

RCA 1800

MICROPROCESSORS

Instruction Manual for the RCA COSMAC Micromonitor CDP18S030

MPM-218

Suggested Price \$5.00

**Instruction Manual
for the RCA COSMAC
Micromonitor CDP18S030**



**Solid
State**

Buenos Aires • Hamburg • Liege • Madrid • Mexico City • Milan
Montreal • Paris • Sao Paulo • Somerville NJ • Stockholm
Sunbury-on-Thames • Taipei • Tehran • Tokyo

Information furnished by RCA is believed to be accurate and reliable. However, no responsibility is assumed by RCA for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of RCA.

Trademark(s) Registered®
Marca(s) Registrada(s)

Copyright 1978 by RCA Corporation
(All rights reserved under Pan-American Copyright Convention)

Printed in USA/1-78

Foreword

When the Micromonitor is used with a COSMAC Development System CDP18S005 and the COSMAC Floppy Disk System CDP18S805, its capability can be considerably enhanced by utilization of the Micromonitor Operating System CDP18S831. The Micromonitor Operating System (MOPS) includes the MOPS diskette CDP18S830, a UART Interface Module CDP18S508, and a connecting cable CDP18S511. MOPS provides an extended set of Micromonitor-type commands that allow the user to conveniently switch Micromonitor commands and responses to and from a variety of peripherals. It also provides several commands which allow a degree of automation in system debugging and testing. The utilization of the Micromonitor Operating System is described in the **RCA COSMAC Micromonitor Operating System (MOPS) CDP18S831 Users' Guide, MPM-222.**

This manual assumes that the reader has a good working knowledge of the CDP1802 microprocessor, such as can be obtained from the **User's Manual for the CDP1802 COSMAC Microprocessor, MPM-201.** If the Micromonitor is used with a development system, such as the Evaluation Kit or COSMAC Development System, the appropriate manual for those systems should also be studied before the Micromonitor is used.

The RCA CDP18S030 Micromonitor is a self-contained, powerful debugging tool for use with any system based on the CDP1802 Microprocessor. It permits in-circuit debugging in real time so that both hardware and software problems can be efficiently screened. The Micromonitor is interposed between the system under test and the system's CDP1802 CPU, giving the user control of both hardware interfaces and execution of the user program. The Micromonitor, although controlled by its own internal microprocessor, uses the microprocessor, power supply, clock, memory, and other components of the system under test to run the user program. Thus, it is not an emulator, but a monitor of system performance. It has been designed to induce a minimal effect on the system under test and provides a reliable measure of true system performance.

A repertoire of simple commands that can be exercised from a self-contained keyboard or a standard external terminal provides control of program execution and hardware interfaces. This command repertoire together with a system of user prompts for providing positive feedback on system operation gives the user a powerful tool for the detection of system problems.

Table of Contents

	Page
Foreword	3
Introduction	7
Installation and Initial Operation	9
Before Connecting the Micromonitor	9
Setup	9
Power Sequencing	10
Display Prompt	10
Further Checks	11
Micromonitor Self-Test Card	11
Notes on Use with CDS CDP18S004	12
Notes on Use with CDS II CDP18S005	12
Control Keys and System Operation	13
Basic Operating Controls	13
Cursor and Data Entry ($\leftrightarrow$)	13
Shift Key (SH)	13
Memory Operations (M)	13
Register Operations	14
Register Mode (R)	14
Registers X and P (X-P)	14
Interrupt Enable and Register T (IE-T)	14
Data Flag and Register D (F-D)	15
Q Output Line (Q)	15
Program Run Modes	15
Real Time Run (\$P)	15
Single-Instruction Cycle (\$N)	15
Single Machine Cycle (\$S)	16
Breakpoints	16
Manual Break (BK)	16
Break Conditions (BC)	16
Break Response	17
Data Logging (LOG)	17
Control of External Signals	17
Wait (WAIT)	17
Clear (CLR)	18
Interrupt (INT)	18
Direct Memory Access (DMAI) (DMAO)	18
Reset Request (RR)	18
Inhibit Request (IR)	18
Flag Lines (EF1) (EF2) (EF3) (EF4)	18
Input Mode (IN)	18
Output Mode (OUT)	19
Control of External Options	19
External Memory (EXM)	19
Terminal Option (10) (30) (120)	19
Parameter Pass Feature	19
Operation of Micromonitor from Terminal	21
Command Syntax	21
Memory Operations	22
Register Operations	23
R Register	23
D Register	23
DF Flag	23
X Register	24
P Register	24
IE Flag	24
T Register	24
Q Flag	24

	Page
Program Run Modes	24
Real-Time Run	24
Single Instruction	25
Single Cycle	25
Breakpoints	25
Set Break Conditions	25
Clear Break Conditions	25
Data Log	26
Control of External Signals	26
Wait	26
Clear	26
Interrupt	26
DMAIN	26
DMAOUT	26
Reset Requests	27
Inhibit Requests	27
Flag Lines	27
Input	27
Output	27
Additional Control Commands	27
External Memory	27
Micromonitor Keyboard	27
Terminal Operation	27
Break Response	27
Micromonitor Hardware	29
How the Micromonitor Executes Instructions	30
Effects of System Clock	30
How the Micromonitor Gains Control	30
Register Save and Restore Operations	31
Instruction and Cycle Counting	31
Control of External Signals	32
External Interfaces to the Micromonitor	32
External Break Input	32
External Memory	33
Terminal Interface	34
Crystal Socket	34
Specifications	35
Example Session	37
Appendix A – RCA Micromonitor Keyboard Command Summary	39
Appendix B – RCA Micromonitor Terminal Command Summary	40
Appendix C – RCA COSMAC Microprocessor CDP1802 Instruction Summary	41
Appendix D – Logic Diagrams for RCA COSMAC Micromonitor CDP18S030	49
Appendix E – Connector Pin Lists	60
Appendix F – Transformer Connections for 115- or 230-Volt AC Operation	61
Appendix G – Instructions for Converting a Model 33 Teletype Terminal from Half- to Full-Duplex Operation and from 60-mA to 20-mA Operation	62
Appendix H – Data Terminal-Micromonitor Connection Details	63
Appendix I – COSMAC Micromonitor Operating System (MOPS) CDP18S831	65
Appendix J – Operation of a Typical Developmental System	66

Introduction

The CDP18S030 Micromonitor offers a complete set of debugging aids for CDP1802 systems. With it, one can examine and alter a system's memory and all the internal registers of the CPU, break on nine different conditions, run programs in three different modes, and control external signals to the CPU.

The Micromonitor can be operated from its built-in keyboard or from an external ASCII terminal. In its simplest configuration the monitor is connected only to the system under test as shown in Fig. 1. The operator can then control the monitor from the built-in keyboard.

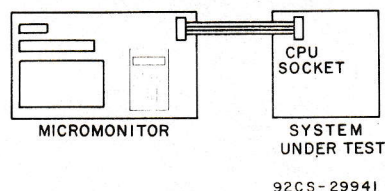


Fig. 1 – Configuration for simple keyboard operation.

If a data terminal is available, it also can be used to control the monitor. In this way, as shown in Fig. 2, the operator can retain a hard-copy record of the debugging session. Without making any changes in

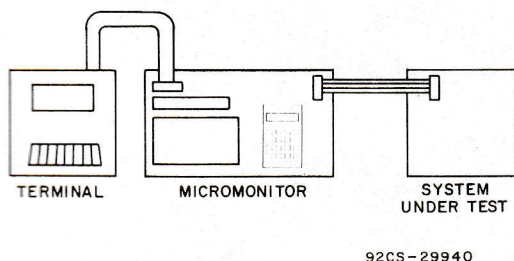


Fig. 2 – Connection of external data terminal for Micromonitor control.

its connection to the Monitor, the terminal can be shared by another system. Fig. 3 for example, shows how a COSMAC Development System may also use the terminal.

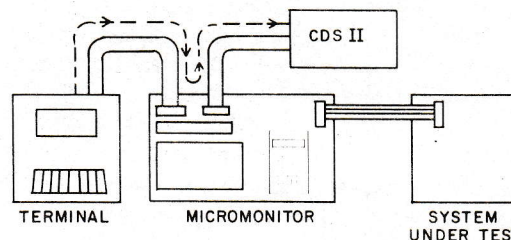


Fig. 3 – Data terminal shared by Micromonitor and CDS.

Whenever the Micromonitor is being operated by the built-in keyboard (as opposed to by the terminal), the signals from the terminal are fed through the Micromonitor to the CDS. As shown in Fig. 4, the terminal can even be shared by the System Under Test. Refer to Appendix H.

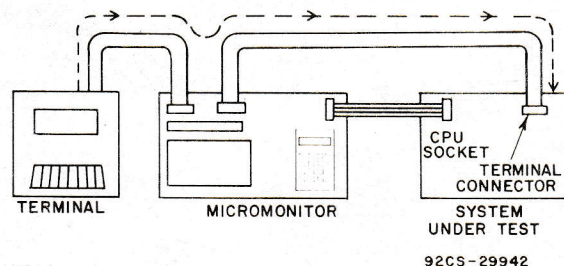


Fig. 4 – Data terminal shared by Micromonitor and system under test.

For some applications, such as production testing, it may be desirable to issue a predetermined set of commands to the Micromonitor. In this case another system can be used to control the Micromonitor via

the terminal interface, as shown in the block diagram of Fig. 5. For such applications, the Micromonitor Operating System (MOPS) CDP18S831 would be a

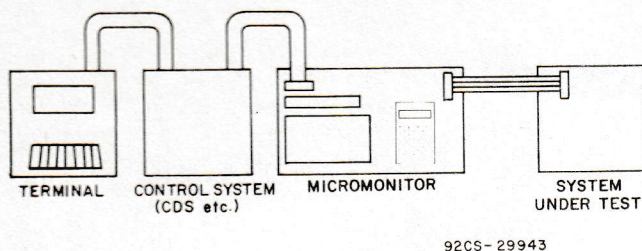


Fig. 5 — Micromonitor operated by a control system consisting of hardware such as the COSMAC Development System II with Floppy Disk option and using the software of the Micromonitor Operating System (MOPS) CDP18S831.

helpful adjunct (See Micromonitor Operating System (MOPS) CDP18S831 Users' Guide, MPM-231 and Appendix I. Also, see Appendix J, Operation of a Typical Developmental System.)

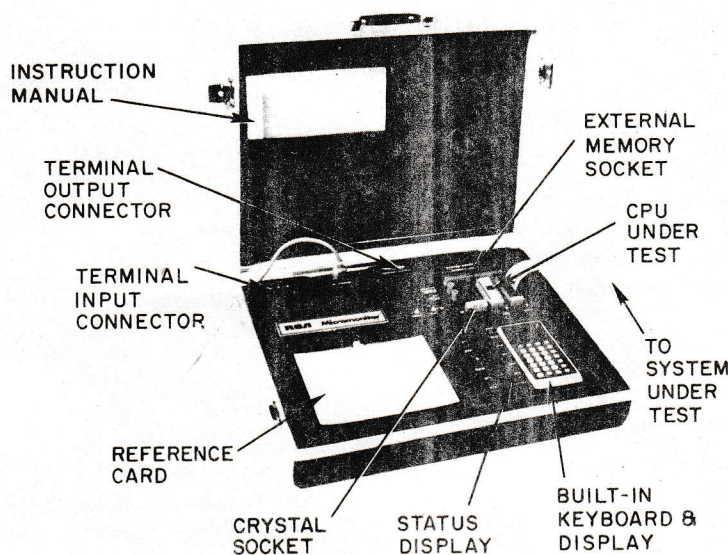
The many different configurations for Micromonitor usage make it applicable in the laboratory for design, in the field for maintenance, or in the factory for production testing. In addition, with its single-cable connection and integral carrying case it is a highly portable instrument.

The advanced features of the Micromonitor allow it to be incorporated into a system at any stage of system development. The external Memory socket (see Fig. 6) allows memory to be added to the system under test via the Micromonitor. For example, a ROM system under development could be checked by running the program from a CDP18S205V1 4-kilobyte RAM module plugged into the Memory Socket.

The Micromonitor has three run modes, \$P, \$N, and \$\$\$. \$P mode runs the system in real time. \$N mode allows the operator to specify the number of instructions to perform. \$\$\$ mode accepts a number of cycles to execute. Neither \$N nor \$\$\$ runs in real time.

Other features of the Micromonitor are:

- Two modes of data logging
- Control of I/O devices
- Tactile-response keyboard with LED display
- Terminal Interface:
 - EIA or TTY
 - 110, 300, 1200 baud
- Tracking power supply over range of 4 to 11 volts
- 110/220 volt 50/60 hertz operation
- Quick-reference abbreviated instructions
- Self contained
- Microprocessor controlled
- Minimal effect on system under test
- Self-test card for verification of Micromonitor operation



92CS-29652

Fig. 6 — Major elements of RCA COSMAC Micromonitor CDP18S030.

Installation and Initial Operation

This Chapter describes how the Micromonitor is connected for testing a CDP1802 Microprocessor-based system and gives the steps to be used for initial operation.

Before Connecting the Micromonitor

A few simple tests should be made on the hardware of the system under test before it is hooked up to the Micromonitor. Only a few signals are required by the Micromonitor to allow it to exercise the system under test. Although other problems may cause the Micromonitor to read or write data incorrectly, the presence of the signals listed below will indicate to the Micromonitor that the CPU under test is obeying its instructions.

1. Check V_{DD} (pin 40) and V_{CC} (pin 16) for appropriate voltage and V_{SS} (pin 20) for ground.
2. Check the clock (pin 1) to make sure that it is running.
3. Check that TPA (pin 34) and TPB (pin 33) are being generated. TPB should always be present when the clock is running unless the processor is in the WAIT mode. TPA should also be present except in the WAIT mode or unless the processor is idling in the LOAD mode. A convenient way to get both pulses is to put the processor in the LOAD mode and generate continuous DMA-OUT requests (pins 2, 3, and 37 all low).
4. Check the State Code lines (pins 5 and 6) for normal operation.

Setup

With all power off, remove the CPU from the system under test (SUT). Solder a 40-pin socket in place of the CPU if one is not already there. The Micromonitor is connected to the SUT through the 40-conductor cable supplied. Pin 1 is denoted by a notch in the corner of the cable terminations. On the sockets of the Micromonitor, pin 1 is adjacent to the handle. Be sure to observe proper polarities when installing the connector and CPU.

Install the CPU in the Micromonitor socket labeled CPU as shown in Fig. 7. Install one end of the cable in the Micromonitor socket labeled CABLE and the other end in the empty CPU socket of the system under test. It is recommended that the crystal of the SUT also be moved to the Micromonitor, particularly for systems operating at higher frequencies. The crystal is inserted into the 14-pin socket labeled CRYSTAL on the Micromonitor, one pin on each side of the socket. The crystal selection switch should be put in the IN position to activate the socket. If the crystal or system clock remains on the SUT, the selector switch should be in the OUT position. For additional information, see the Chapter Hardware under the subheading "Crystal Socket".

Make any connections desired to the External Break Input, Memory Disable Output, or insert any external memory card before turning power on. Also, connect terminal inputs and output, if desired, at this time. See the Chapter Hardware for a discussion of these options.

Table I summarizes the set-up steps.

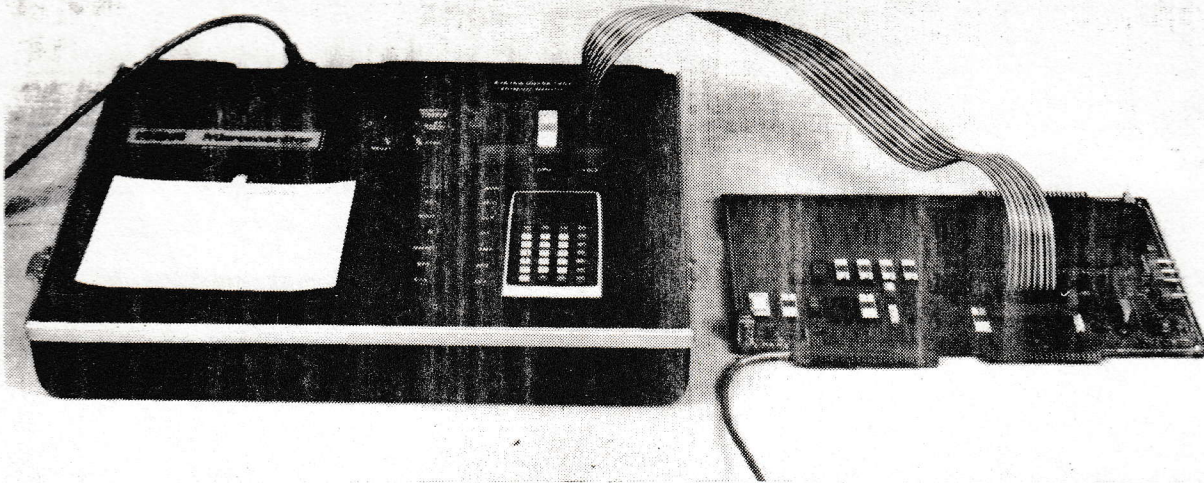


Fig. 7 — Micromonitor connected to system under test.

TABLE I. Check List of Steps Required Before
The Micromonitor is Turned On.

1. The 40-conductor cable should be connected properly at both ends. Observe polarity!
2. The CDP1802 of the system under test should be in the Micromonitor CPU socket.
3. Any connections to the External Break Input, the Memory Disable Output, or the Memory Card Socket should be made.
4. The Crystal IN/OUT switch should be in the desired position. If the switch is in the up position (IN), a crystal or oscillator should be connected to the 14-pin socket of the Micromonitor.
5. If a data terminal is used, it should be connected to the Terminal connector. If the Terminal Output is used, it should also be connected.
6. Both the Micromonitor and the system under test should be plugged in.
7. Do not apply power to the system under test unless the Micromonitor is ON.

Power Sequencing

To avoid damage to either system, the system under test should never be on while the Micromonitor is off. First, turn on the Micromonitor power switch, located near the fuseholder; then, turn on the system under test.

To turn the power down, reverse the above procedure. First turn off the system under test and

then turn off the Micromonitor. Never insert or remove IC's with power on. It is recommended that the Micromonitor and SUT both be plugged into a common AC power distribution box.

Display Prompt

As a check that the connections are properly made, after the Micromonitor and the SUT are turned on it is helpful to push the Reset switch to display the eight decimal points, as shown in Fig. 8. (The center digit

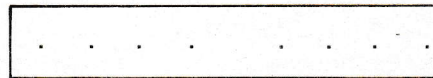


Fig. 8 — Micromonitor display prompt.

of the display is not used by the Micromonitor.) This display is the general user prompt and means that the Micromonitor is running and is ready to accept a command. The eight decimal points will reappear as a prompt if an erroneous key is pressed, causing the Micromonitor to exit that command mode, or whenever the Micromonitor is ready to accept another command.

If the prompt does not appear after the Reset switch is pushed, power should be shut off and the connections rechecked. If the problem persists, refer to the following sections "Further Checks" and "Micromonitor Self-Test Card."

Further Checks

After the Micromonitor is Reset, it extracts the CPU registers before issuing its prompt. If any of the signals mentioned in the section "Before Connecting the Micromonitor" are missing, the Micromonitor will "think" that the SUT is running very slowly. It will wait for the CPU to execute a command and may never display a prompt. If the prompt does not appear, disconnect the Micromonitor, recheck the signals listed above, and then, if needed, check the Micromonitor stand-alone operation with the self-test card as described in the next section.

If the prompt does appear, make two simple tests to verify the integrity of the data bus and address bus. First, examine Q by pressing SH and then Q. A zero or 1 should be displayed which should agree with the indicating LED on the Micromonitor. Toggle the display with the ENT key. If a number other than zero or one appears, there is a problem in the data bus.

After the data bus has been verified, the address bus can be checked with the \$\$ command. Press \$\$ and enter 0000 as the starting address. Press ENT and 0000 should be displayed in the address field and the data byte at that location in the data field. If the address is not 0000, there is a problem with the address lines. For example, an address of 0404 indicates that address line A2 is shorted high. Note that the high-order address bits being displayed come from an internal latch in the Micromonitor. They are not necessarily what the SUT sees. A short on the output of the SUT's address latch will not be detected by this method.

The same procedure should also be applied with an address FFFF to detect any bits shorted to ground and with alternating bit patterns AAAA and 5555 to detect bit lines shorted together.

Micromonitor Self-Test Card

A self-test card is supplied with the Micromonitor so that the user can verify some basic Micromonitor operations. The card is set up to provide terminations for the 40-wire connector cable to simulate signals from a system. The connections are shown in Fig 9. Although there is no memory on the test card, the data bus is pulled high through 22-kilohm resistors so that any "memory location" read will appear to contain an FF. Because FF is a valid instruction code for the CDP1802 (Subtract Memory Immediate), the processor can actually fetch and execute this instruction from the non-existent memory.

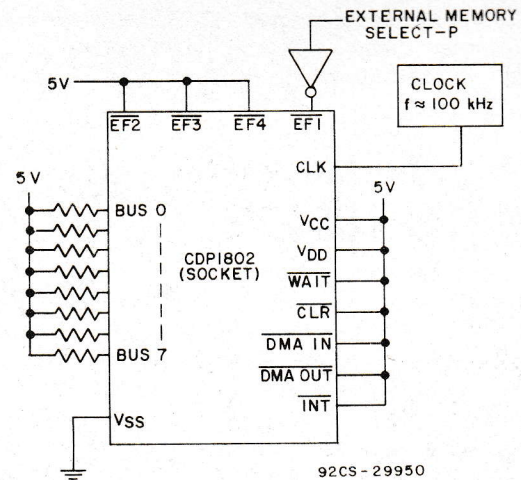


Fig. 9 — Connection diagram for Micromonitor self-test card.

The test card should be inserted in the External Memory socket (with the component side facing front) and the free end of the 40-conductor cable plugged into the DIL socket. Be sure to observe proper polarity. The Crystal In/Out switch should be in the Out position. A CDP1802 known to be good must be in the Micromonitor CPU socket. Turn power on and push the Reset switch. The decimal point prompt should appear. If it does not, the CPU or the Micromonitor itself may be defective. Also, check to make sure that the oscillator on the test card is operating. Then, recheck all of the signals listed in the section "Before Connecting the Micromonitor." Finally, substitute a second CDP1802 in the Micromonitor CPU socket. The test card operates at 5 volts so that either a CDP1802D or CDP1802CD may be used. If the Micromonitor still fails to display a prompt, it is probably defective.

Once the prompt does appear, the Q mode operation described previously should be tried. In that mode, a substantial portion of the Micromonitor circuitry is exercised. Noting that SMI #FF to be executed is equivalent to an ADI #01 will provide some ideas for further tests. For example, the break-point logic can be tested by setting a break on READ from location FFFF. First, set D to 00 and then do \$P0000. After about a second, the Micromonitor should break. Check that P=0, D=00, and R(0)=0000. Try other break addresses as well.

The External Memory Select Signal can also be tested because it is connected to EF1. Set EXM to 1 and see if the indicator LED comes on. Also, program a break on EF1 and test it.

If the Micromonitor does not operate properly with a CPU known to be good, it is suggested that the unit be returned for repair. It is not advisable to attempt

to troubleshoot the Micromonitor itself because it is a very complex system and any modifications to the system or unauthorized repairs may void the warranty.

Notes on Use with CDS CDP18S004

When the Micromonitor is used with the CDS CDP18S004, two items should be taken into consideration. First, it should be noted that the RESET button in the CDP18S004 disables the CDP18S004 oscillator circuit. Consequently, if this oscillator circuit is being used as the system clock, after power on or after the RESET button has been pressed, the RUN U or RUN P button must be pressed to make sure that the system has a clock.

Second, it should be noted that when RUN U or RUN P is pressed, a DMA OUT request is generated in the CDP18S004 system as part of the normal starting method. If the Micromonitor is in control, this DMA request will remain until the system is released with a \$P, \$N or \$S command with IR=0. The user, therefore, prior to beginning a debug

session, should clear out this request by executing a \$N1 command.

Notes on Use with CDS II CDP18S005

When the Micromonitor is used to debug a program running on the CDSII CDP18S005 and use is to be made of the Micromonitor's External Memory option, such as, for example, the 4-kilobyte RAM module CDP18S205V1, the following modifications must be made to the CDS II to allow the External Memory Select signal to disable the memory space in the CDS (See "External Memory" on page 33 of the Micromonitor Hardware section).

1. Disconnect the +V line at pin 15 of each of the four CDP1856's (U3, U4, U5, and U6) on the CPU module CDP18S102.
2. Connect the External Memory Deselect-N line (pin A of the Micromonitor's External Memory Interface Connector) to pin 15 of the four CDP1856's in place of +V on the CPU module CDP18S102.
3. Connect the External Memory Deselect-N line (pin A of the Micromonitor's External Memory Interface Connector) to pin X of the Address Latch and Bank Select module CDP18S206 (MBDS-N).

Control Keys and System Operation

In this Chapter, the various keys used for control or data entry are described along with the various modes of system operation.

Basic Operating Controls

Cursor and Data Entry (↔)

Most command modes display more than one number at a time. For example, the Memory Address mode displays a memory address and its contents. The cursor is a set of decimal points indicating which number or field is being addressed by the keyboard. To move the cursor, press the ↔ key. This key causes the cursor to toggle between the two fields of the display. If there are more than two, the cursor moves to the left, wrapping around to the right-hand when it gets to the end.

To change a number on the display, move the cursor to that field and punch in the new number. The new number is shifted into the right of the field. The left-most number is lost.

Example

8.0.0.0. C 4

In memory mode, M(8000) contains a C4. If "1" is pressed, the display changes to

0.0.0.1. F 8

Note the right hand number changed to show the data at the new location, 0001.

To enter data into the address field, a push of the ↔ key moves the cursor to the data field.

0 0 0 1 F.8.

When a "1" is pressed, the following appears.

0 0 0 1 8.1.

The new data (81) is written to location 0001 when the ENT button is pressed, as described in the next section. Prior to the pressing of ENT, only the display is altered, not the actual data location.

Note: All numbers are to be in hex (address, data, or operation numbers) for all Micromonitor commands.

Shift Key (SH)

Many of the Micromonitor's keys perform two separate functions. To use the second function, the shift key (SH) must be pressed prior to pressing the function key. The display acknowledges the SH key by showing a special prompt - every other decimal point lighted. In the shift mode, pressing a control key causes the Micromonitor to use the shifted function (function printed on the case). This mode can be used only if the Micromonitor is in control.

Memory Operations (M)

The Memory Mode is used to examine and modify memory. To enter the memory mode press the M key. An address and the data at that address is displayed as shown in Fig. 10. This mode can be used only if the Micromonitor is in control.

0. 0. 0. 0.	F 8
ADDRESS	DATA
FIELD	FIELD
M(0000) = F8	

Fig. 10 – Memory mode display.

The initial address displayed is usually 0000 when this mode is entered. If a new address is entered, the data field key automatically shows the contents of that location. Pressing the ENT key enters the data in the display field into the displayed address and then increments the address by 1.

To enter a new number into a memory location, move the cursor to the address field and enter the desired location. Then, move the cursor to the data field and enter the desired byte. Finally, press the ENT key. The address is automatically incremented and the contents of the new location shown.

To read the contents of a memory location, simply enter the desired address (cursor in address field). Pressing the ENT button increments past any location without changing the byte there if the display is not altered. If the displayed data byte is erroneously changed, pressing $\leftrightarrow$ before ENT causes the actual data at the displayed address to reappear.

Register Operations

Register Mode (R)

The Register mode is used to examine and modify any of the 16 "R" registers of the CPU. To enter the Register mode, press R. A register number and the contents of that register is displayed as shown in Fig. 11. This mode can be used only if the Micromonitor is in control.

1 2 3 4	3.
CONTENTS	REGISTER
OF	NUMBER
REGISTER	

Fig. 11 – Register mode display.

The cursor appears under the register number which initially is usually 0. Operation in this mode is similar to the Memory mode. To read the contents of a register, enter the desired register number. Pressing the ENT key increments the register number and the contents of the new register are displayed.

To change the contents of a register, move the cursor to the data field and key in the desired number. Pressing the ENT key causes the new number to be written to the designated register. If $\leftrightarrow$ is pressed before ENT, the original contents will reappear.

Registers X and P (X-P)

In this mode the value of X and P can be examined and modified. To enter the X-P mode, press the X-P key. X and P are then displayed as shown in Fig. 12.

1	0.
X = 1	P = 0

Fig. 12 – X-P mode display.

The cursor starts under P. To change either X or P, move the cursor to the correct field, enter the new value, and press ENT. If $\leftrightarrow$ is pressed before ENT, the original value returns and remains unaltered. This mode can be used only if the Micromonitor is in control.

Interrupt Enable and Register T (IE-T)

In this mode IE and T can be examined and modified. To enter this mode, press SH and then IE-T. IE and T are displayed as shown in Fig. 13. The

0	1. 0.
IE = 0	T = 10

Fig. 13 – IE-T mode display.

cursor starts under T. Either value can be changed by moving the cursor to the appropriate digit, entering the desired value, and pressing the ENT key. Only values 0 or 1 are accepted for IE. If $\leftrightarrow$ is pressed before ENT, the original contents will reappear. This mode can be used only if the Micromonitor is in control.

Data Flag and Register D (F-D)

In this mode the value of DF and D of the CPU can be examined and altered. To enter this mode press F-D. DF and D are displayed as shown in Fig. 14. The

1	B. 2.
---	-------

DF = 1 D = B2

Fig. 14 – F-D mode display.

cursor starts under D. Either value can be changed by moving the cursor to the appropriate digit, entering the desired value, and pressing the ENT key. Note that DF can only have the value 1 or 0. An attempt to enter any other value into DF is ignored. If ↔ is pressed before ENT, the original contents will reappear. This mode can be used only if the Micromonitor is in control.

Q Output Line (Q)

The Q Mode is used to examine and modify the Q flip-flop of the CPU. To enter Q mode press SH then Q. To toggle the value of Q, press ENT. An example of the Q display is given in Fig. 15. This mode does

	1.
--	----

Q = 1

Fig. 15 – Q mode display.

not simply look at the Q output pin of the CPU, but does a conditional branch on the state of Q as part of a complex software procedure. If the CPU is defective or a system fault is preventing proper execution of instructions, a value other than 0 or 1 may be shown for Q, providing useful debugging information. This mode can be used only if the Micromonitor is in control.

Program Run Modes

Programs may be run either in real time from a specified address, in single/multiple instruction cycles, or in machine cycles. These modes of operation are discussed next. These modes can be used only if the Micromonitor is in control to start execution. Execution of these commands causes the Micromonitor to relinquish control.

Real Time Run (\$P)

The \$P key is used to start the system under test running in real time. After the \$P key is pressed two fields are displayed as shown in Fig. 16. The starting

0 0 0 1	
-----------------	--

BREAK STARTING
POINT ADDRESS
COUNTER

Fig. 16 – Real-time run display.

address field initially is blank. If no address is entered, the system under test is started, after ENT is pressed, without changing X, P, or R(P). This mode is used to continue the microprocessor from its present state.

If a starting address is specified, X and P are set to 0 and R(0) to the starting address. It is important to remember that X, P, and R(P) are not changed until the ENT key is pressed starting the system under test.

The left field is the break point counter. After each occurrence of a break point the count is decremented and checked. If the result is not zero, the system under test is automatically continued. The result is that the system under test stops at the n^{th} occurrence of a break point. The default value is 1 which stops execution at the first occurrence of a break point. See next section "Breakpoints" for information on setting break points.

Between breaks, the system runs in real time as determined by the SUT's clock. When a break is generated, however, the SUT will require additional time while the breakpoint counter is decremented and checked and the log updated. See Fig. 32 in Chapter Micromonitor Hardware under subheading "Instruction and Cycle Counting."

Single-Instruction Cycle (\$N)

The single-instruction cycle mode is used to execute a fixed number of instructions. When the \$N key is pressed two fields are displayed as shown in Fig. 17.

0 0 0 1	
-----------------	--

INSTRUCTION STARTING
COUNTER ADDRESS

Fig. 17 – Single-instruction cycle display.

The display and operation are similar to the \$P command. The starting address is initially blank. If no address is entered, the SUT continues (after ENT is pressed) from its present state without changing X, P, or R(P) until the desired number of instructions has been executed. If a starting address is specified, X is set to 0, P to 0, and R(0) to the starting address.

In this mode, the SUT is halted after every instruction, the instruction counter decremented, and checked. If the counter is not zero, the system is restarted. Note that the SUT does not operate in real time in this mode.

Break conditions are enabled in this mode so that if a break occurs, the SUT is halted and will not complete the programmed set of instructions. If this situation occurs, the break conditions are displayed rather than the \$N completion display of all zeros and decimals. Refer to the next section "Breakpoints" for a discussion of break conditions.

Single Machine Cycle (\$S)

The single machine cycle mode is used to execute a fixed number of machine cycles. To enter this mode press the \$S key. Two fields are displayed as shown in Fig. 18. The display and operation are similar to the

0	0	0	1	.	.	.	.
MACHINE				STARTING			
CYCLE				ADDRESS			
COUNTER							

Fig. 18 — Single machine cycle display.

\$N and \$P commands. With no starting address specified, the SUT continues from its present state, when ENT is pressed, until the desired number machine cycles have been executed. If a starting address is specified, X is set to 0, P to 0, and R(0) to the starting address.

The SUT is stopped after each cycle and the cycle counter decremented and checked for zero. After the counter reaches zero, the system stops and the memory address and data bus is displayed. Each depression of the ENT switch thereafter will cause one more machine cycle to occur and the new address and data bus state to be displayed. Note that no break conditions are allowed in the \$S mode and that all breaks are cleared when this mode is entered. To exit this mode, Manual Break is pressed. The Micromonitor will halt SUT execution on the next S0 cycle. It should also be noted that all commands which do not require the Micromonitor to be in

control may be executed in this mode. For example, the user might step to a test of EF1 and then set EF1 using the Micromonitor. The \$S display of the address and data bus may be returned by pressing \$S. Pressing ENT results in another machine cycle execution if this display is present. See Chapter "Hardware" for further information on system operation in the \$S mode.

Breakpoints

Manual Break (BK)

A Manual break can be accomplished at any time by pressing the BK key. The display shows the decimal point prompt, indicating that the Micromonitor is ready to accept commands. This facility permits, for example, breaking into a faulty program loop which has no exit. If the location of the loop is unknown, the single-step operations (\$N or \$S) can be used to locate the path.

A Manual break is the preferred way of interrupting a user program because the state of the machine is preserved. The BK key will reset a WAIT or CLEAR signal being generated by the Micromonitor.

Break Conditions (BC)

Break conditions can be set and examined by pressing the BC key. Each digit on the display represents a particular break condition. The conditions are printed on the case right below each digit. A break condition enabled is shown as a '1' and disabled as a '0'. Initially, after the Micromonitor is reset, no break conditions are enabled. Breaks can be enabled on any combination of the following:

- 1) EF1 - break when EF1 goes true ($\overline{\text{EF1}} = V_{SS}$)
- 2) EF2 - break when EF2 goes true ($\overline{\text{EF2}} = V_{SS}$)
- 3) EF3 - break when EF3 goes true ($\overline{\text{EF3}} = V_{SS}$)
- 4) EF4 - break when EF4 goes true ($\overline{\text{EF4}} = V_{SS}$)
- 5) EXT - break when the signal connected to the EXTERNAL BREAK input is a '1' (V_{DD})
- 6) IDL - break when an Idle occurs (3 successive S1 states)
- 7) S3 - break when an Interrupt is acknowledged (S3 state)
- 8) MAS - break on a read/write memory condition at a specified address

For MAS, the display can be one of the following:

- 0 - No memory address stops
- 1 - Break on read

- 2 - Break on write
- 3 - Break on read or write

Break conditions are set by moving the cursor to the position of interest, entering at 1 or 0, and pressing ENT. MAS can be set to 0, 1, 2, or 3.

After ENT is pressed with the cursor under MAS, the display changes so that the address can be entered, as shown in Fig. 19. The address is modified by entering the number and pressing ENT.

0 . 0 . 0 . 0 .

ADDRESS
FOR
MAS

Fig. 19 - Display for MAS address entry.

Breaks normally occur synchronously at the start of the first S0 cycle following the break condition. The break leaves R(P) pointing to the next instruction which would normally be executed after the break condition, so that if a break is set on a branch the user can see which path would be taken. Note that if the microprocessor is executing an IDLE instruction when the break is taken, the next instruction is the idle itself.

Break Response

If the System Under Test is stopped because of a manual (BK) break or completion of a \$N command, the prompt is displayed. If it is stopped because of the completion of a \$N command, all zeros and all decimals are displayed. If it is stopped because of a break condition that has been set, however, the display shows the reason for the break. An example of the display is given in Fig. 20. In this example, the

0 . 1 . 0 . 0 .	0 . 0 . 0 . 0 .
-----------------	-----------------

Fig. 20 - Display from set break condition showing EF2 true.

System Under Test was stopped because EF2 went true.

If two or more break conditions occur simultaneously, they are all displayed as in the example in Fig. 21 where the System Under Test was

0 . 1 . 0 . 0 .	1 . 0 . 0 . 0 .
-----------------	-----------------

Fig. 21 - Display from set break conditions showing EF2 and EXTERNAL BREAK true.

stopped with EF2 and the EXTERNAL BREAK signal both going true.

Only those conditions which actually caused the break are displayed. For instance, if a break is set only for EF1, then the other flags will be displayed as 0's in the break response, regardless of their actual states (high or low).

Data Logging (LOG)

Whenever a break condition is met, or at the completion of an instruction in the \$N mode, the values of D, X, P, and R(P) are recorded. Up to 16 states can be preserved with the latest state displacing the oldest in a push-down stack arrangement. To read out the data log, push SH then LOG. The data log number (zero being the latest state) and the values of D and R(P) are displayed as shown in Fig. 22.

0 .	F 8	0 0 0 1
-----	-----	---------

LOG D R(P)
NUMBER

Fig. 22 - Data log display showing D and R(P).

Pressing ENT increments the data log number, stepping backwards in time. Pressing $\leftrightarrow$ changes the display to show D, X, and P as shown in Fig. 23.

0 .	F 8	1 0
-----	-----	-----

LOG D X, P
NUMBER

Fig. 23 - Data log display showing D, X, and P.

Because the data log is never cleared, but only updated, old information beyond a certain point may not be relevant to a particular debugging operation. It can only be examined when the Micromonitor is in control.

Control of External Signals

Wait (WAIT)

This command can be used to set the WAIT signal to the CPU and may be used at any time. This signal is logically OR'd with the WAIT signal from the SUT. Note that the display shows the Micromonitor-generated signal, which is not necessarily the actual state of the CPU's WAIT status. To determine the actual status, look at the LED display of the WAIT line that appears on the panel. A '1' in the display or the LED ON indicates that the condition is asserted (WAIT pin of the CPU is at V_{SS}).

To enter the command mode press SH then WAIT. The display will be a '1' (WAIT asserted) or '0' (WAIT not asserted by the Micromonitor) in the right-hand digit. Pressing ENT will toggle the display.

Clear (CLR)

This command is used to set the CLEAR signal to the CPU and may be used at any time. Like WAIT, it is logically OR'd with the signal from the SUT. Again, a '1' in the display or the LED ON indicates that CLEAR is asserted (CLEAR pin of the CPU is at V_{SS}).

To enter the command mode press SH then CLEAR. The display shows the Micromonitor-generated signal (1 = asserted, 0 = not asserted) in the right-hand digit. Pressing ENT toggles the display.

Interrupt (INT)

To generate an interrupt request, press SH then INT. This signal is also OR'd with externally generated requests. This command can be exercised whether or not the Micromonitor is in control. The LED on the panel indicates the status of the interrupt request line. If ON, an interrupt request is asserted (INTERRUPT pin of the CPU is at V_{SS}).

Pressing SH, INT resets any pending DMAIN or DMAOUT requests from the Micromonitor. The interrupt request is reset by any of the following (refer to subsequent sections):

- 1) S3 state
- 2) DMAIN or DMAOUT request
- 3) Reset Request
- 4) Inhibit Request

Direct Memory Access (DMAI) (DMAO)

To generate a DMAIN or DMAOUT request, press SH then either DMA request key. These signals are OR'd with externally generated requests. These commands can be generated whether or not the Micromonitor is in control. Here again, the LED's on the panel indicate the status of the request lines (ON for DMA asserted).

A DMAIN request resets on pending Interrupt or DMAOUT requests from the Micromonitor. Likewise, a DMAOUT cancels Interrupt and DMAIN signals. The DMA requests are reset by any of the following:

- 1) S2 state
- 2) Interrupt request
- 3) Another DMA
- 4) Inhibit Request
- 5) Reset Request

Reset Request (RR)

Pressing SH and then RR resets any pending Interrupt or DMA requests from the Micromonitor. This command may be exercised whether or not the Micromonitor is in control.

Inhibit Request (IR)

Interrupt or DMA requests from the SUT can be enabled or disabled with this command. To activate this mode, press SH then IR. This display shows a 0 or 1 in the right-hand digit. A '1' indicates that requests are inhibited from the SUT; a '0' means that they are enabled. The ENT switch toggles between the two modes.

Note that requests from the Micromonitor are not inhibited by this command mode. This command may be exercised whether or not the Micromonitor is in control.

Flag Lines (EF1) (EF2) (EF3) (EF4)

To generate a signal on a flag line, press SH followed by the desired EF flag key. A '1' indicates that the Micromonitor is asserting that flag line. (EF is forced to V_{SS}). These signals are OR'd with signals generated by the SUT. The Micromonitor cannot, therefore, set a flag to '0' if the SUT is generating a '1'. The display shows the Micromonitor-generated signal. Check the LED on the panel for the actual flag line status. The display is toggled by the ENT button. These commands may be exercised whether or not the Micromonitor is in control.

Input Mode (IN)

The Input Mode is used to read data from the System Under Test's input devices. To enter the Input Mode press SH then IN. To read a device, enter the desired device number (1-7) and press ENT. The data received from that device is displayed. This operation does not alter the state of the CPU or the SUT. Data is not altered in memory nor are any CPU registers affected. This command merely allows the user to see what will be transmitted if a particular input instruction is issued. A sample display is given

in Fig. 24. This command can only be used if the Micromonitor is in control.

3 .		4	2
DEVICE		DATA	
NUMBER		BYTE	

Fig. 24 — Sample input mode display showing device number and data byte.

Output Mode (OUT)

Data can be sent to a specified output device using the Output Mode. This mode is entered by pressing SH then OUT. The display has three data fields as shown in Fig. 25, the data byte, the output device number, and the memory address.

6	A	4	0 . 1 . 8 . 0 .
DATA	DEVICE	MEMORY	
BYTE	NUMBER	ADDRESS	

Fig. 25 — Three data fields for output mode display.

Initially, the address and device number fields are zero. The data field initially shows the contents of the selected address. Move the cursor and enter values, as required, for the data byte to be output, for the device number of the output port ($N=1-7$), and for the memory address from which the data is to be sent. Pressing ENT causes the output operation to occur.

If no data is specified, the byte already at that memory location is sent. If data is specified, the memory address must, of course, be a RAM location. The byte already there is saved by the Micromonitor and restored after the output operation, thus leaving the CPU and system memory unaffected by the operation. This command can only be used if the Micromonitor is in control.

Control of External Options

External Memory (EXM)

External Memory can be substituted for the memory of the SUT with this command. To enter this mode, press SH then EXM. The display shows a '1' or '0'. A '1' deselects the memory of the SUT and selects the memory in the external memory socket. The ENT button toggles the value of the display. Do not set EXM to 1 unless the SUT memory is deselected either by removal or by proper connection

to the memory disable jack. This command should be used only when the Micromonitor is in control. Refer to the Chapter on Hardware for more information on adding external memory.

Terminal Option (10) (30) (120)

The Micromonitor always starts with the built-in keyboard in control after it is Reset. Control can be transferred to an external terminal by pressing SH followed by the key selecting the desired baud rate (10, 30, or 120 characters per second). A decimal point prompt is then typed. Control is returned to the keyboard when a \$K is typed on the terminal. The decimal point prompt will again be displayed. Refer to the Chapter on Hardware for a discussion of terminal interfacing. Control can be transferred only if the Micromonitor is in control.

A summary of all keyboard commands is given in Appendix A.

Parameter Pass Feature

A built-in feature of Micromonitor operation permits examination of X, R(X) and MR(X); or P, R(P) and MR(P); or any MR(N) with only three (four for X) key strokes. When the Memory mode is entered, the initial address is 0000 unless the register mode has been accessed since the last memory mode selection. In that case the address will come from the register last selected. For example, to show the contents of MR(3), do the following:

Press Key	Display	
R	0 1 8 0	0.
3	4 1 7 2	3.
M	4.1.7.2.	2 1
	R3	MR(3)

Similarly, when the Register mode is entered, the initial register number is 0 unless the X-P mode was accessed since the last register mode selection. In that case, the register number will be the value in X or P depending on the position of the cursor when the X-P mode was left. For example, to examine MR(P), do the following ($P=5$):

Press Key	Display
X-P	1 5.
R	0 2 7 8 5.
M	0.2.7.8. F 9
	R(P) MR(P)

Press Key	Display
X-P	1 5.
↔	1. 5
R	1 7 0 4 1.
M	1.7.0.4. 3 5
	R(X) MR(X)

Or, to see MR(X), do the following (X=1):

Operation of Micromonitor from Terminal

The Micromonitor can be operated from an ASCII data terminal instead of the built-in keyboard, if a hard copy record of the debugging session is desired. The data terminal can be connected in several different configurations, as discussed in the first Chapter Introduction. Details on terminal hookups and interfacing are given in the Chapter on Hardware under the subheading "External Interface to the Micromonitor."

When the Micromonitor is powered on* and/or Reset, its keyboard is activated and the external data terminal deactivated. To activate the data terminal, first a baud rate must be selected. Above the bottom left three keys of the keyboard are the numbers 10, 30, and 120, giving the available rates in characters per second. The terminal interface is initially set up for full-duplex operation. This set-up can be changed to half-duplex operation with the !CH command, as explained later in this Chapter under "Additional Control Commands." Press the SH key followed by the 10, 30, or 120 key, as appropriate. The terminal will type a carriage return, line feed followed by a period. The period is the Micromonitor's prompt, indicating that it is in control and is ready for a command.

The Micromonitor issues a prompt at the completion of every command. To indicate when the Micromonitor is **not** in control a '%' prompt is substituted for the period.

The following material describes how to activate a particular operation on command. Details on the

* **Caution:** The System Under Test should never be on while the Micromonitor is off.

operation of each command is given in the Chapter Control Keys and System Operation.

Command Syntax

Commands to the Micromonitor from an external data terminal are of three categories:

1. **Modify Commands**
These commands begin with an ! and change the value of a memory location, register, or otherwise perform an action.
2. **Interrogate Commands**
These commands begin with a ? and are used to examine the value of a memory location, register, or condition.
3. **Run Commands**
These commands cause execution of a program in one of the various modes.

The general form of a command is shown in Fig. 26.

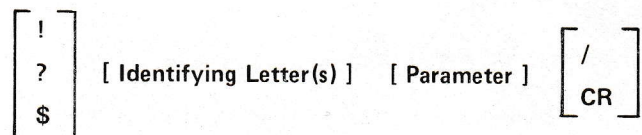


Fig. 26 — General form of terminal command.

Note that all commands begin with either an !, ?, or \$ (except for the manual break function as described later). Any characters before the initial !, ?, or \$ are ignored.

A command that is improperly entered causes the Micromonitor to type a carriage return, line feed, and a '?'.
 ?M230024

All commands can be terminated by either a carriage return (CR) or a slash (/). The slash option saves paper and, in the case of most ? commands, causes the output to be printed on the same line.

All numbers entered through the Micromonitor are to be in hex. If more digits are entered for a parameter than are required, then only the last four (or fewer if required) typed are used. This feature permits the correction of typing errors without aborting the command. For instance

?M230024

is equivalent to

?M0024

Leading zeros are assumed for all digits and are used if too few digits are entered for a parameter. For instance,

?M7

is equivalent to

?M0007

Any command can be aborted by a "Control-C" character generated by holding down the Control key and typing a 'C'. This action causes the Micromonitor to ignore the previous command and issue a prompt. For example, a

?M25(↑C)

does not cause any memory typeout.

In the examples following, characters generated by the Micromonitor are underlined and all user input not underlined. The nomenclature is as follows:

aaaa = address
 dddd = data
 nnnn = number

The symbol Δ is sometimes used for emphasis to indicate a required space in a command.

Memory Operations

The memory interrogation command is of the form

?MaaaaΔnnnn

where aaaa = starting address

nnnn = number of bytes

The default conditions (no value entered) for the parameters are:

aaaa = 0000
 nnnn = 1

The format of the printout is a line beginning with an address followed by data grouped in two-byte blocks. When necessary, new lines are started every 16 bytes, each beginning with the new address. An example is given in Fig. 27.

```
. ?M0000 20 (CR)
0000 1122 7483 5A94 176F 1293 EAD7 6432 5C89;
0010 BEBC 8687 E405 6FDE 953C 1796 538B E98F
. ?M(CR)
0000 11
. ?M1/
0001 22
```

Fig. 27 — Sample printout of memory interrogation response (Characters generated by Micromonitor are underlined).

The ?M command provides the facility for a memory dump in reloadable format.

The memory altering command has the form

!MaaaaΔdd

where aaaa = starting address (default = 0000) and dd... = data byte(s).

Data is entered into memory after each two hex digits are typed. If an odd number of digits is typed, the last digit is ignored and the error indication '?' is displayed. Only the last bytes, therefore, remain unchanged.

The !M command provides two options of line continuation. First, a string of data can be extended from line to line by typing a comma just before the normal CR. (In this case press the CR-LF (carriage return-line feed) keys before beginning a new line.) For example,

```
!M23 56789ABC, (CR) (LF)
DEF0123456, (CR) (LF)
3047 (CR)
```

enters 11 successive bytes beginning at location 0023. Between successive hex pairs while data is being

entered, any non-hex character except the comma (and semicolon, as will be discussed) is ignored. This arrangement permits arbitrary LF's, spaces (for readability), nulls (generated by the utility program or by a time-share system to give the carriage time to return), etc.

As a second optional form of data entry, a string of input data can be terminated by a semicolon (and a CR). The utility program then expects more data to follow on the next line, but preceded by a new beginning address. The line must have the format of an !M command, but with the !M omitted. This option provides the mechanism for reading in a paper tape previously punched out as a result of the ?M command. (Recall the format of multiline ?M outputs discussed above.)

The Micromonitor ignores all non-hex characters following !M allowing CR, LF, and nulls to be inserted in the tape without disturbing the !M command. The semicolon feature allows non-contiguous memory areas to be loaded.

The ?M and !M commands can be used only if the Micromonitor is in control.

Note

Only hex object files should be loaded through the Micromonitor. An attempt to load a complete listing file (that includes the source code) may result in errors.

Register Operations

The ! and ? register commands can be used only if the Micromonitor is in control.

R Register

The sixteen 'R' registers can be examined and altered by means of the ?R and !R commands. They have the format

!Rn hhhh where n = register number 0-F (default n=0)

and hhhh = data (default hhhh = 0000)

?Rn (default n = all)

An example is given in Fig. 28.

The !R command permits loading successive registers in the same way !M does. For example,

```
. !R3 1234 5678, (CR) (LF)
9ABC RF DEF0/
```

```
. ?R1(CR)
5A27
. ?R(CR)
5A27 8E45 18D4 3821 5B33 B760 8A15 0017
5518 0717 34AA 8197 A401 6789 A825 01B9
.
. !R9 1234/
. ?R9/ 1234
. !R/
. ?R0/ 0000
```

Fig. 28 — Sample printout of register examination and command response (Characters generated by Micromonitor are underlined).

loads 1234 into R3, 5678 into R4, 9ABC into R5, and DEF0 into RF.

The other CPU registers and flags can be examined and set using ? and ! commands as shown in examples below.

D Register

Format: !Dhh (default hh = 00)

Execution: Load register D with the data value hh.

Format: ?D

Execution: Print contents of register D.

Example:

```
. !D17/
. ?D/ 17
. !D/
. ?D/ 00
```

DF Flag

Format: !Fb (b = 0, 1 ; default b = 0)

Execution: Set the DF flag to zero or one.

Format: ?F

Execution: Print the value of the DF flag.

Example:

```
. !F1/
. ?F/ 1
. !F/
. ?F/ 0
```


X Register

Format: !Xn (default n = 0)

Execution: Load register X with the register number n.

Format: ?X

Execution: Print the contents of register X.

Example:

```

. !XA/
. ?X/ A
. !X/
. ?X/ 0

```

P Register

Format: !Pn (default n = 0)

Execution: Load register P with the register number n.

Format: ?P

Execution: Print the contents of register P.

Example:

```

. !P3/
. ?P/ 3
. !P/
. ?P/ 0

```

IE Flag

Format: !IEb (b = 0, 1; default b = 0)

Execution: Set the Interrupt Enable flag to zero or one.

Format: ?IE

Execution: Print the value of the Interrupt Enable flag.

Example:

```

. !IE1/
. ?IE/ 1
. !IE/
. ?IE/ 0

```

T Register

Format: !Thh (default hh = 00)

Execution: Load register T with the hex number hh.

Format: ?T

Execution: Print the contents of register T.

Example:

```

. !TA/
. ?T/ 0A
. !T/
. ?T/ 00

```

Q Flag

Format: !Qb (b = 0, 1; default b = 0)

Execution: Set the Q flip-flop to zero or one.

Format: ?Q

Execution: Print the value of the Q flip-flop.

Example:

```

. !Q1/
. ?Q/ 1
. !Q/
. ?Q/ 0

```

Program Run Modes

The program run modes are controlled by the \$P, \$S, and \$N commands as outlined below. They are used when the Micromonitor is in control to start SUT execution and result in the Micromonitor relinquishing control.

When a break occurs, the break symbol '->' is typed, followed by break condition(s) that caused it. Break conditions stop execution in the \$N mode. In the \$N and \$S modes a -> also appears after the number of instructions or machine cycles have been executed (and no break conditions encountered). The ->, however, is not followed by anything in this case, so that this response is distinguishable from that caused by a break condition.

For details on the exact sequence of operations evoked by each of these commands, see Chapter on Hardware.

Real-Time Run

Format: \$Paaaa:hhhh

Execution: Start program running from location M(aaaa) with P=X=0. Micromonitor regains control at the hhhhth occurrence of a break condition.

Default: aaaa = R(P) ; X, P unmodified
hhhh = 1

Thus, typing \$P(CR) causes a program to continue from the present state until the next break condition is encountered.

Single Instruction

Format: \$Naaaa:hhhh

Execution: Execute hhhh instructions starting at M(aaaa) with P=X=0.

Default: aaaa = R(P) ; X, P unmodified
hhhh = 1

Data is logged on each instruction cycle.

Examples:

SN:4 continues from present location for 4 instructions

SN3 executes the instruction at M(0003)

Single Cycle

Format: \$Saaaa:hhhh

Execution: Execute hhhh machine cycles starting at M(aaaa) with P=X=0.

Defaults: aaaa = R(P) ; X, P unmodified.
hhhh = 1

State code, address, and data bus contents are printed for each machine cycle

Examples:

1) Execute 4 machine cycles starting from address 0

```

. $S0:4
0 0000 F8
1 0001 01
0 0002 A3
1 8100 01
->
.

```

2) Continue for 3 cycles

```

. $S:3
0 0003 F8
1 0004 13
0 8101 B3
->
.

```

Note that if the state code is S0 on completion of the count, then the SUT will be allowed to complete the execution cycle(s) before the break occurs. The execution cycle(s) will not be printed.

Breakpoints

Breakpoints can be set, cleared and examined using the commands described in the following sections.

Set Break Conditions

Format: !BScccc....

Execution: Set break conditions specified

Default: cccc... = all

Each break condition is requested by a 1-digit mnemonic as follows:

- 1 - Break on EF1 true ($\overline{EF1} = V_{SS}$)
- 2 - Break on EF2 true ($\overline{EF2} = V_{SS}$)
- 3 - Break on EF3 true ($\overline{EF3} = V_{SS}$)
- 4 - Break on EF4 true ($\overline{EF4} = V_{SS}$)
- E - Break on External Break Input = V_{DD}
- I - Break on Idle (3 or more S1 states)

S - Break on Interrupt Response (S3 state)

R(aaaa) - Break on Read from address aaaa

W(aaaa) - Break on Write to address aaaa

(default aaaa = last value specified, initially 0000)

Only one address can be set. An additional break condition can be set at any time or the break address changed without affecting break conditions already set. (Note that running \$S mode clears all break conditions.) Once set, a break condition can only be reset by the !BC command. An address is always printed whether a Read/Write break is enabled or not when the break conditions are questioned by use of the ?BS command.

Clear Break Conditions

Format: !BCcccc... (see break codes above)

Execution: Reset break conditions specified

Default: cccc... = all

Example:

```

. !BS/
. ?BS/ 1234EISRW (0000)
. !BSR(70)/
. ?BS/ 1234EISRW (0070)
. !BC/
. ?BS/ (0070)

```


Data Log

The data log can be read for any number of entries with the ?L command, but only when the Micromonitor is in control.

Format: ?Lh (h = 0 - F)

Execution: Causes printout of h + 1 entries in the log

Default: h = all

Example:

```

. ?L/
P R(P) X D
0 000A 0 03   latest entry
0 8008 3 02
0 0008 0 00
0 800C 0 01
0 0008 0 02
0 800C 1 00
0 0009 0 02
0 4009 0 08
0 0009 1 0D
0 0008 0 01
0 000C 0 06
0 0001 0 02
0 000D 0 02
0 4000 1 05
0 400A 0 02
0 8009 0 01   oldest entry
. ?L4/
P R(P) X D
0 000A 0 03
0 8008 3 02
0 0008 0 00
0 800C 0 01
0 0008 0 02

```

Control of External Signals

The commands listed below can be used whether or not the Micromonitor is in control except for the input and output commands which require the Micromonitor to be in control.

Wait

Format: !Wb (b = 0 = not asserted; b = 1 = asserted; default b = 0)

Execution: Assert (or release) a Micromonitor-generated signal to the WAIT line of the CPU. This signal is logically OR'd with the WAIT signal from the SUT.

Format: ?W

Execution: Print the status of the Micromonitor-generated signal to the WAIT line of the CPU.

Example:

```

. !W1/
. ?W/ 1
. !W/
. ?W/ 0

```

Clear

Format: !Cb (b = 0 = not asserted; b = 1 = asserted; default b = 0)

Execution: Assert (or release) a Micromonitor-generated signal to the CLEAR line of the CPU. This signal is logically OR'd with the CLEAR signal from the SUT.

Format: ?C

Execution: Print the status of the Micromonitor-generated signal to the CLEAR line of the CPU.

Example:

```

. !C1/
. ?C/ 1
. !C/
. ?C/ 0

```

Interrupt

Format: !I

Execution: Asserts Interrupt request ($\overline{\text{INTERRUPT}} = 0$) from the Micromonitor. It is reset by an S3 state, the !RR command, the !IR command, the !DO command, or the !DI command.

DMAIN

Format: !DI

Execution: Asserts DMAIN request ($\overline{\text{DMAIN}} = V_{SS}$) from the Micromonitor. It is reset by an S2 state, the !RR command, the !IR command, the !I command, or the !DO command.

DMAOUT

Format: !DO/ $\left[\begin{array}{c} \text{half-} \\ \text{duplex} \\ \text{mode} \end{array} \right]$ or !DO/ $\left[\begin{array}{c} \text{full-} \\ \text{duplex} \\ \text{mode} \end{array} \right]$

Execution: Asserts DMAOUT request ($\overline{\text{DMAOUT}} = V_{SS}$) from the Micromonitor. It is reset by an S2

state, the !RR command, the !IR command, the !I command, or the !DI command.

Reset Requests

Format: !RR

Execution: Resets any pending Micromonitor-generated INTERRUPT, DMAIN, or DMAOUT signals.

Inhibit Requests

Format: !IRb (b = 0 = allow; b = 1 = inhibit; default b = 0)

Execution: Inhibit (or allow) SUT-generated DMA and Interrupt requests to the CPU. Execution of this command also clears current Micromonitor-generated DMA and Interrupt requests, but does not inhibit future ones from the Micromonitor.

Format: ?IR

Execution: Print the status of the request inhibiting logic.

Flag Lines

Format: !EFf b (f = flag number = 1 - 4; default f = all; b = 0 = not asserted; b = 1 = asserted; default b = 0)

Execution: Assert (or release) a Micromonitor-generated signal to the specified flag input(s) of the CPU. These signals are logically OR'd with the respective EF signals from the SUT.

Format: ?EF (all)

Execution: Print the status of the Micromonitor-generated signals to the EF lines of the CPU.

Example:

```

. ?EF/ 0 0 0 0
      ↑   ↑
      EF1 EF4

. !EF 1
. ?EF/ 1 1 1 1
. !EF/
. ?EF/ 0 0 0 0
. !EF 1/
. ?EF/ 1 0 0 0

```

Input

Format: ?Ip (p = 1 - 7)

Execution: Read the selected input port

Default: None - a number must be entered for p.
Usable: Micromonitor in control only.

Example:

. ?I1/ 7E

Output

Format: !Oaaaa:hh:p

Execution: Send data byte hh to output port p via M(aaaa).

Default: aaaa = 0000

hh = present contents of M(aaaa).

p = 1

Usable: Micromonitor in control only.

Additional Control Commands

External Memory

Format: !EMb (b = 0 disable external memory
b = 1 enable external memory)

Execution: Enable or disable external memory.

Default: b = 0

Format: ?EM

Execution: Print the status of external memory enable/disable logic.

Micromonitor Keyboard

Format: \$K

Execution: Disable terminal interface and return control to the built-in keyboard.

Usable: Micromonitor in control only.

Terminal Operation

Format: !CH

Execution: Set the terminal-Micromonitor communications link for half-duplex operation.

Usable: Any time.

Format: !CF

Execution: Set the communications link for full-duplex operation. Operation is initiated in this mode.

Usable: Any time.

Break Response

Whenever a break condition occurs, the Micromonitor prints a -> followed by the one-letter mnemonic describing which break caused the halt. They are the same mnemonics used when break

conditions are set with the addition of an 'M' which is the response to a manual break:

- 1 - break on EF1
- 2 - break on EF2
- 3 - break on EF3
- 4 - break on EF4
- E - break on External Break Input
- I - break on Idle
- S - break on S3 cycle

R(aaaa) - break on Read at M(aaaa)
W(aaaa) - break on Write at M(aaaa)
M - Manual Break

When the Micromonitor halts the SUT at the completion of a \$N or \$S command, the -> is also printed but without any conditions appearing. It is followed by a carriage return and "." prompt.

A summary of the terminal commands is given in Appendix B.

Micromonitor Hardware

The Micromonitor is actually a small computer system built around its own CDP1802. It contains 6 kilobytes of ROM and 256 bytes of RAM. The various components of the Micromonitor are simply different I/O devices to its CPU. Fig. 29 is a representation of the Micromonitor in block diagram form.

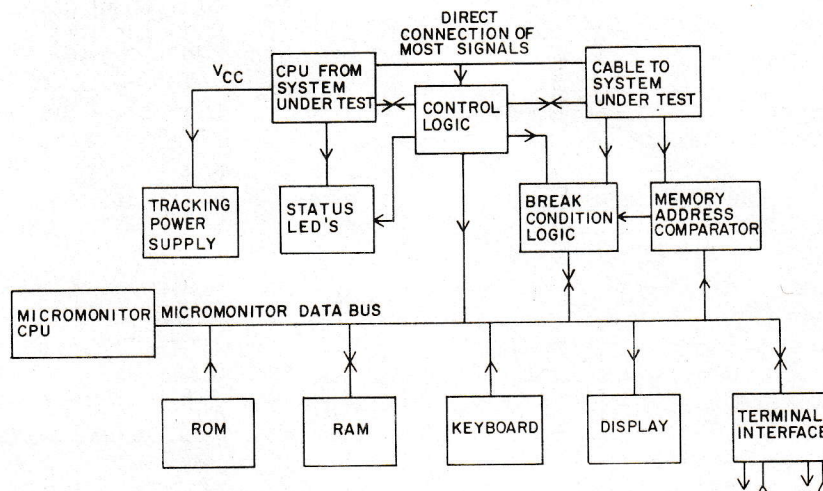
The keyboard is an input device to the Micromonitor. Logic circuits in the Micromonitor decode the key matrix into a unique 8-bit representation for each key and a "key-depressed" signal. Pressing two keys simultaneously produces an invalid key code. The keyboard is debounced by software routines in the Micromonitor ROM's.

The multiplexed seven-segment display uses two output ports. One port holds segment information, the other port selects which digit is to be displayed.

Hex-to-seven-segment conversion and display multiplexing are handled by utility routines in the Micromonitor.

The terminal interface uses a CDP1854 UART. The UART appears to the Micromonitor as two input devices and two output devices from which it reads and writes data and status. "Don't care" bits in the status byte are used to select the speed of the UART clock.

The heart of the Micromonitor is represented by the block labeled "Control Logic" which functions as a combination of several different input and output devices. Through this control logic, the Micromonitor can control state code lines, request lines, MRD, EF flags, etc. It can also "force" the CPU of the system to execute instructions and can capture data from the CPU's bus.



92 CM - 29948

Fig. 29 — Block diagram of Micromonitor CDP18S030.

The Break Condition Logic has two output and two input ports. The Micromonitor sends to the output port the break conditions which it wishes to enable and reads through the input port the condition which caused the system to break. The Memory Address comparator constantly compares the address bus with the programmed stop address. If the two are equal, it sends a signal to the Break Condition Logic.

Fourteen LED's are provided so that the operator can examine the status of the Micromonitor and the System Under Test. These lights indicate what the CPU "sees" when it starts to run its program.

A Tracking Power Supply samples the V_{CC} voltage from the SUT. All of the Micromonitor logic runs at the same voltage as that of the System Under Test.

How the Micromonitor Executes Instructions

The Micromonitor performs its tasks by making the CPU System Under Test execute instructions. For example, if the Micromonitor "wished" to set $M(1234)=56$, it would make the CPU execute the following sequence:

```
LDI #12
PHI R2
LDI #34
PLO R2
LDI #56
STR R2
```

The advantage of using the SUT CPU is that the Micromonitor requires no hardware between the system CPU and the memory because it accesses the memory via the CPU.

The Micromonitor operates in two basic modes: Micromonitor In Control (user program stopped) or Micromonitor Not In Control. When the Micromonitor is in control, the LED labeled "MON" will be on and the System Under Test will be halted. When the SUT is started, the "MON" LED will go off and the Micromonitor will no longer be in control.

Because the Micromonitor makes the system under test execute instructions in order to perform some of its commands, some commands can only be used when the monitor is in control. For example, the case of writing to memory just described could only be done when the Micromonitor is in control. On the other hand, setting EF4 to "1" requires no internal access to the CPU and can therefore be done even when the Micromonitor is not in control.

Effects of System Clock

Because the Micromonitor uses the SUT CPU to perform many operations, its response speed is directly related to the SUT clock. For instance, a memory readout via the ?M command will be extremely slow if the SUT is operated from a very slow clock. Similarly, extraction of CPU registers, which is performed whenever a break occurs, is fixed by the SUT clock and can result in prolonged response to a break condition if the clock is very slow.

When the SUT loads memory through the terminal interface, its clock must be sufficiently fast to allow data bytes coming in at the maximum transmission rate to be written to memory without overlap problems. No first-in-first-out buffering is provided by the Micromonitor. For the available baud rates of 110, 300, and 1200 the minimum permissible SUT clock frequency is given in Table II.

TABLE II. Relation of Minimum Clock Frequency and Baud Rate

Terminal Baud Rate	Minimum SUT Clock Frequency (kHz)
110	0.9
300	2.6
1200	11.8

How the Micromonitor Gains Control

When a break occurs, the Micromonitor waits until the next S0 cycle and then puts an IDLE Instruction (00) on the bus. At the same time it gates off the State Code lines, $\overline{MRD}$, and $\overline{MWR}$ to the system under test. While the Micromonitor is in control, therefore, the CPU appears to be in a long S0 cycle to the system under test. The Clock, TPA, and TPB are not gated off so that a system with dynamic memory can use these signals. When the Micromonitor relinquishes control, it issues a "phony" DMAOUT request to move the processor out of the IDLE state. The resultant S2 cycle is invisible to the SUT.

Thus, when the Micromonitor takes control, it does so on an S0 cycle. The leading edge of S0 is, in fact, used as the enabling signal. This scheme affects the way the Micromonitor handles single cycles in the \$\$ mode. In this mode, the WAIT line to the CPU is asserted at the middle of each machine cycle while the bus status is read. At the completion of the programmed number of cycles, a break condition is set, the WAIT line is released, and the CPU is allowed to go to the next S0 cycle, at which time the break is taken and the Micromonitor regains control.

This sequence means that if the \$\$ terminates on an S0 cycle, the SUT automatically advances to the next S0. Thus, the next printout is of the S0 to which the system has advanced and not of the intervening S1 cycle(s). Also, the \$\$ mode is the only time the TPA and TPB pulses are halted.

DMA cycles represent a special case. The Micromonitor breaks on S2 by shutting off the DMA request lines to cause an S0 with a concomitant break. On start up, an S0 and S1 are first executed before a return to S2 can occur. A summary of the states involved for the \$\$, \$N, and \$P modes is given in Fig. 30.

Register Save and Restore Operations

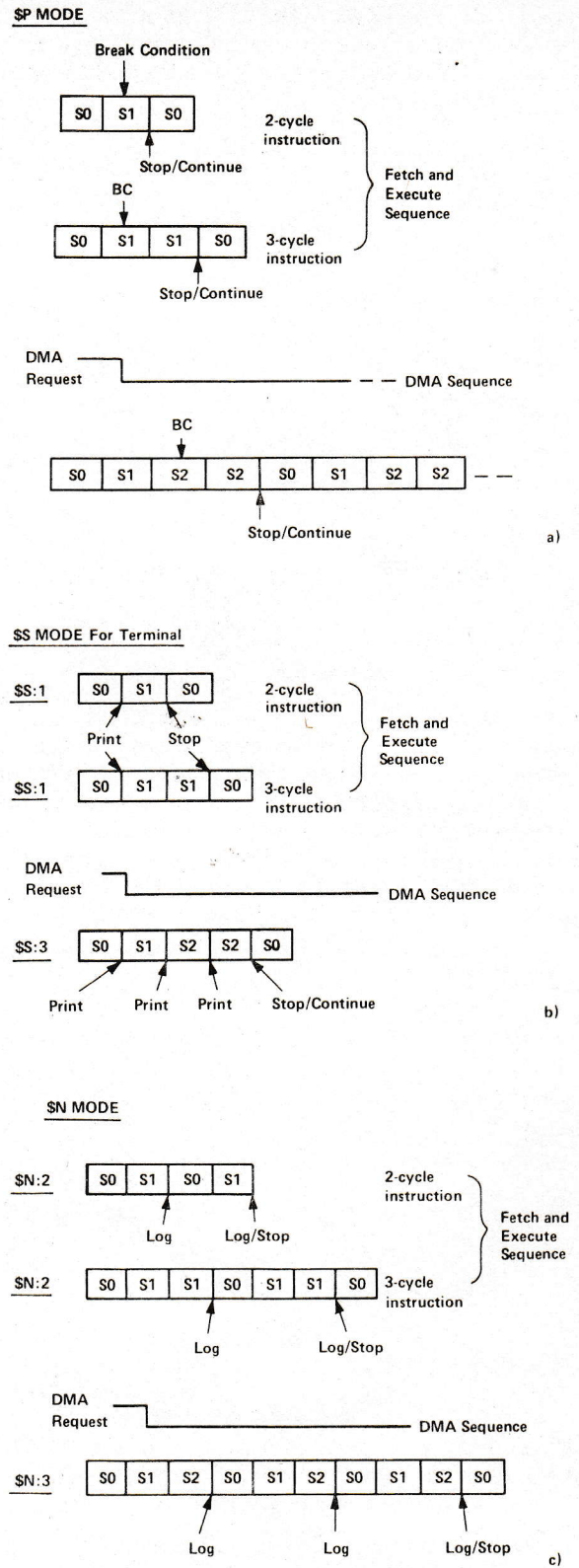
Whenever the Micromonitor gains control, it extracts the contents of the CPU's internal registers and stores them in its RAM. Conversely, whenever the Micromonitor transfers control back to the user program, the contents of this save area are restored as the initial CPU state. When registers are altered via the ! commands, the memory save area is actually what is being altered. The Micromonitor saves registers whenever a break (manual or programmed) occurs or whenever the Micromonitor is reset. However, a manual break is the recommended way in which a user should manually interrupt a program. The Reset operation performs extra functions, such as clearing all break conditions, returning control to the Micromonitor keyboard, etc.

Note that \$P, \$N, or \$\$, without a specified address, merely causes restoration of the memory save area to continue a user program. These commands should not be used without an address to transfer control to the user program the first time unless the user has already properly initialized the save area with an appropriate series of ! commands.

When the Micromonitor restarts the SUT, it restores the SUT CPU registers and issues a DMAOUT request to get the CPU out of the IDLE state. The value of R0 is pre-decremented by the Micromonitor, so that it has the correct value after the DMA operation. Both the save and restore operations are "invisible" to the SUT.

Instruction and Cycle Counting

In the \$\$ cycle-counting mode, the WAIT line of the CPU under test is asserted after each instruction while the Micromonitor checks the count against the programmed number of cycles. This sequence slows down effective execution to 5.56 milliseconds per machine cycle for any microprocessor clock frequency



92CM-30303

Fig. 30 — Operating state sequences.

a) \$P mode; b) \$\$ mode; c) \$N mode.

above 1440 Hz. For lower clock frequencies, refer to Fig. 31. In addition, if the Micromonitor is being operated from a terminal, the time required to print the state code, memory, and data bus statuses must be added to each cycle.

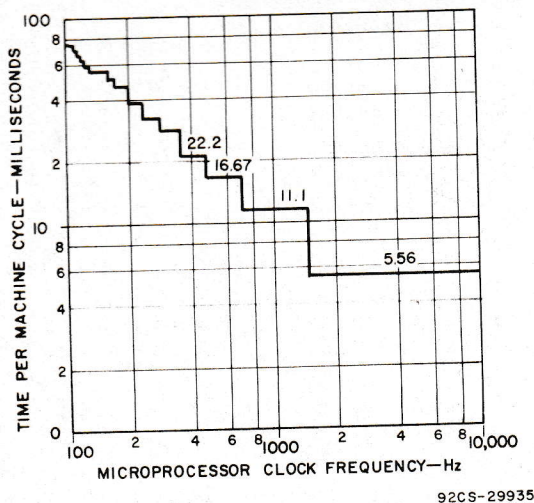


Fig. 31 — Micromonitor \$S mode slow down as a function of clock frequency.

In the \$N mode, a break is performed after each instruction during which the entire state of the machine is saved and later restored, and the Micromonitor checks to see if the programmed number of instructions has been executed yet. To determine the slow-down factor for the \$N mode, refer to Fig. 32.

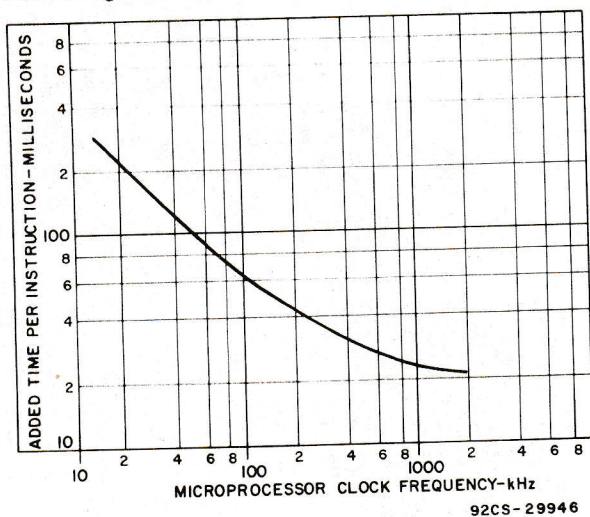


Fig. 32 — Micromonitor \$N mode slow down as a function of clock frequency.

Note again that after a break of any kind occurs the program counter is left pointing at the next instruction that would normally be executed. Thus, if an idle is encountered in the \$N mode, the idle is fetched and executed repeatedly until the count is completed.

Control of External Signals

The Micromonitor can set external signals to the CPU (WAIT, CLEAR, INT, DMAOUT, DMAIN, and EF1 through EF4). These signals are logically "OR'd" with signals from the SUT, as shown in Fig. 33a and b. The request signals (DMAOUT, DMAIN, and INT) can be controlled fully by the Micromonitor because signals from the SUT can be gated off as shown in Fig. 33b. The flag signals are not fully controlled. For example, there is no way to set flag input to a '1' if the SUT is generating a 0.

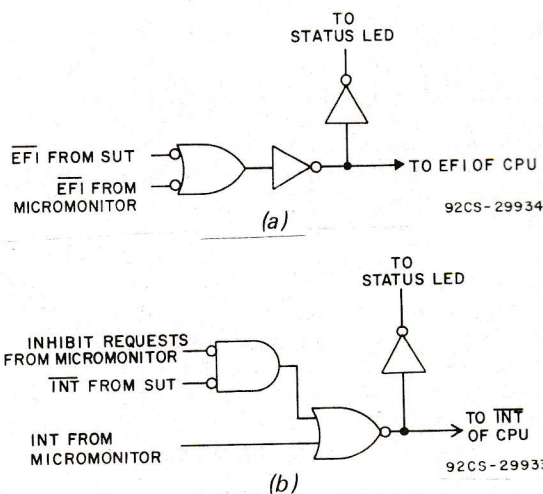


Fig. 33 — External signal inputs to the CPU.
(a) Interface for CPU lines; (b) Interface for DMAOUT, DMAIN, INT, CLEAR, and WAIT lines.

Note that the LED's show the OR'd result that is the actual signal on the input pin. The status LED's indicate whether or not a signal is asserted (on if asserted; off if not asserted).

External Interfaces to the Micromonitor

External Break Input

To assure a clean break, the external break input signal should be a positive-going pulse with a substantial pulse width. A dual banana jack is provided (black = V_{SS} , red = input) for inputting the signal.

External break is a standard CMOS input with a 22-kilohm resistor to V_{SS} subject to the constraints of good CMOS design practice. It should swing from V_{SS} to V_{CC} of the SUT and its maximum voltage must not exceed V_{CC} . Values for the minimum pulse width (50% point) are given in Table III.

TABLE III. External Break Minimum Pulse Width Parameters

5 Volts		10 Volts	
Typical	Worst Case	Typical	Worst Case
560 ns	1250 ns	270 ns	530 ns

External break triggers a flip-flop which samples the break conditions. A shorter pulse (or glitch) could trigger the break but the cause for the break would not be properly recorded.

External Memory

External memory or more complex boards containing, perhaps, memory paging logic can be plugged into the external memory socket. The pins and signal names for this socket are given in Appendix E. The socket is made to accept a CD-P18S205V1 4-kilobyte RAM module directly. However, because all 16 address lines are brought to this connector, custom-built memory circuits can also be used. In general, user-installed memory should have read and write cycles at least as fast as the SUT memory that it is designed to replace or supplement. The Micromonitor power supply is capable of supplying approximately 400 mA to the external module at a voltage matching the SUT voltage.

The signal EXTERNAL MEMORY SELECT-P is high when the external memory module is to be enabled. This signal should be gated with address decoding (if required) into the chip-select inputs of the memory devices on user-designed cards. This signal is controlled by the External Memory key and the EM command as described in the two preceding Chapters under the subheads "Control of External Options" and "Control of External Signals."

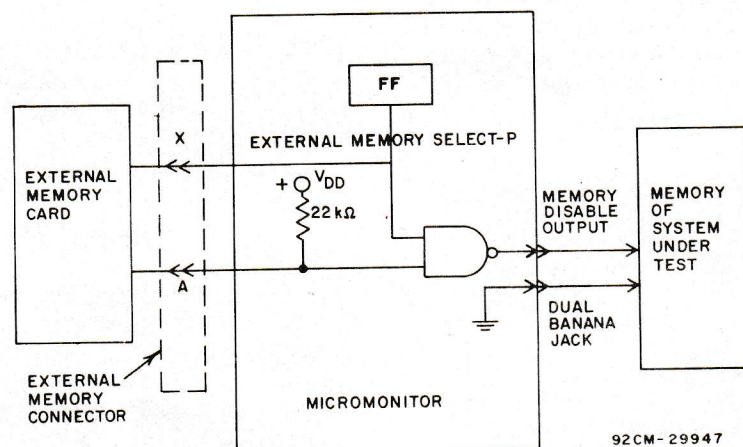
A dual banana jack on the Micromonitor provides a deselection signal for the SUT memory (red = signal, black = V_{SS}). When external memory is selected, this line, labeled Memory Disable Output, goes low. It must be appropriately wired into the selection logic for the SUT memory to disable it on a low signal.

An additional input, memory connector Pin A, allows hardware on the custom memory card to decide whether to select its memory or the memory which is resident to the system under test. If this input, normally pulled high (logical 1) by a 22-kilohm resistor, is driven low (logical 0), the signal at the banana jack goes high (regardless of the value of External Memory Select-P). See Fig. 34 for a block diagram of the interface. This signal can be used for memory paging where, for example, a defective area of SUT memory can be replaced by external memory or RAM substituted for ROM and vice-versa.

Fig. 35 shows a diagram for a typical external memory card designed for paging in the first 4-kilobytes of SUT memory. As shown, the external card is active for pages 0-3, 6, 7, 10-15 (a page = 256 bytes); the SUT memory is used for the remaining pages 4, 5, 8, and 9. The decoding circuitry is set up to enable the external card only in the first 4-kilobytes of memory. By moving the connection to the CD4515 decoder, the card can be reassigned to any 4-kilobyte block in the 64-kilobyte addressable memory field.

There are a number of other possibilities including a ROM-based test program on the external card.

Note that the $\overline{MRD}$ line is not gated off to the SUT memory when that memory is disabled so that any I/O ports using the $\overline{MRD}$ line is still properly addressed. Likewise, $\overline{MWR}$ is not gated off either.



92CM-29947

Fig. 34 — Block diagram of system memory and external memory interface.

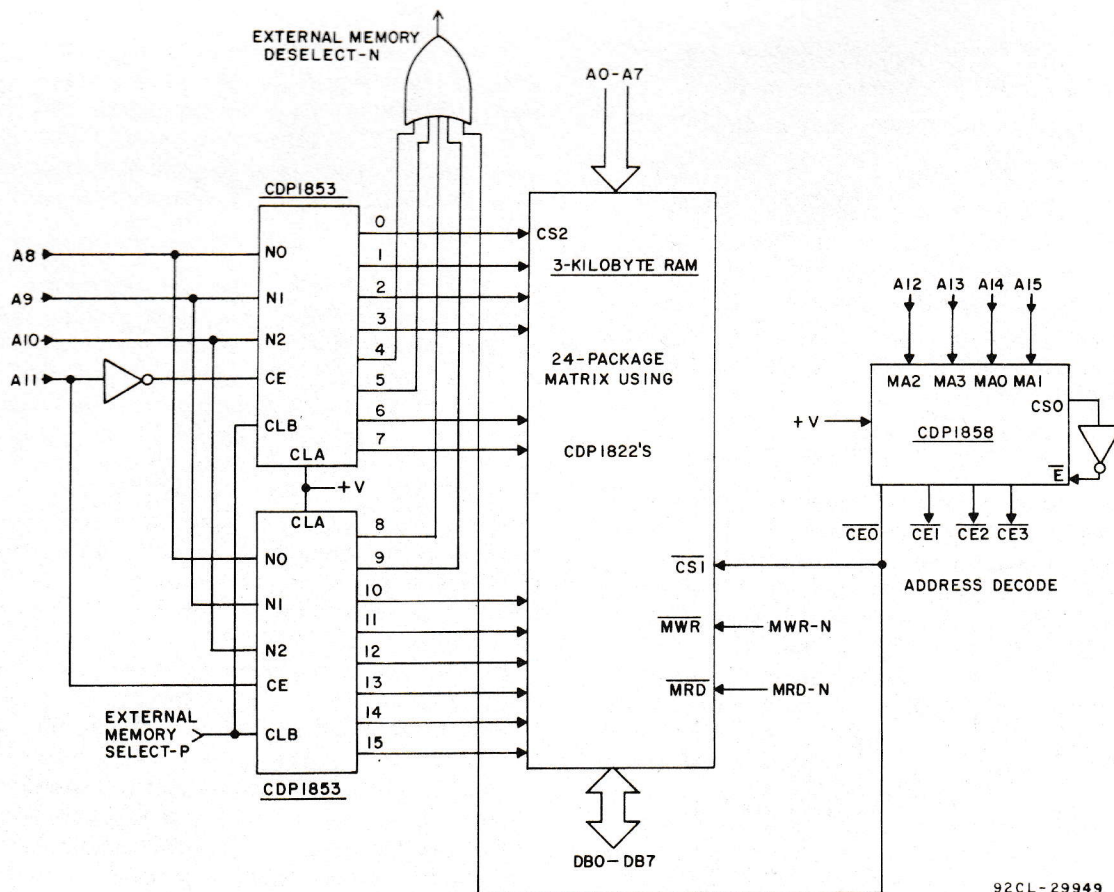


Fig. 35 — Diagram of typical paging external memory card.

Terminal Interface

The Micromonitor interfaces to an external data terminal via a serial ASCII code using either a 20 mA or EIA RS232C standard electrical interface. The data format is given in Fig. 36. The Micromonitor generates even parity, but does not check for parity on incoming data. It generates two stop bits when set for operation at 10 characters per second (110 baud), but only 1 stop bit for operation at 30 (300 baud) or 120 (1200 baud) characters per second.

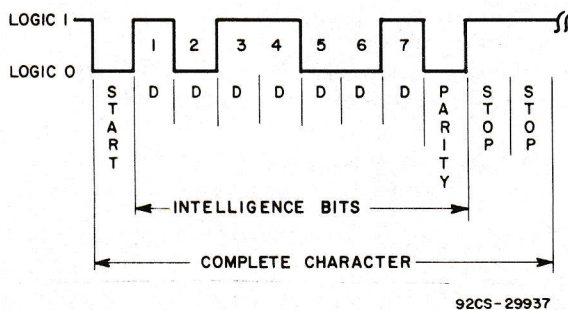


Fig. 36 — ASCII serial data format.

Pin assignments for the terminal input connector are shown in Fig. 37. When the Micromonitor is in the keyboard mode, signals from the terminal input are buffered and fed through to the terminal output connector, which is also equipped for both 20 mA and EIA interfaces. Terminal output connector pin assignments are given in Fig. 38.

The interface is initialized for full duplex operation. It can be changed to half duplex by the !CH command, as described in the Chapter Operation of Micromonitor from Terminal under the subheading "Additional Control Commands."

Crystal Socket

If the SUT is being run by a crystal, it may be necessary to move the crystal closer to the microprocessor to eliminate the effects of the interconnect cable. For this purpose a 14-pin zero-insertion connector is provided on the Micromonitor immediately adjacent to the CPU connector. The crystal should be plugged in as shown in Fig. 39 using one pin on each side of the socket and with the XTAL Switch in the IN position.

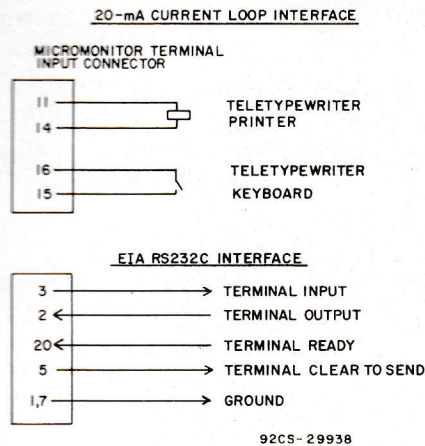


Fig. 37 — Terminal input connector (J1) pin assignments.

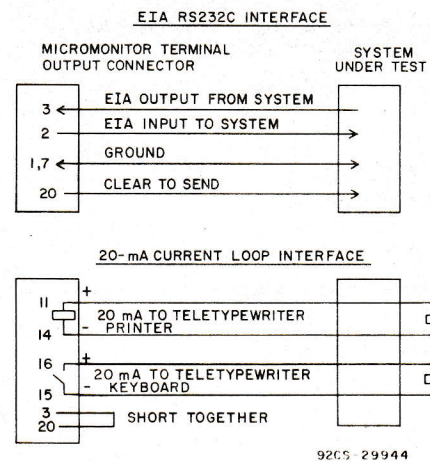


Fig. 38 — Terminal output connector (J2) pin assignments.

The lower side of this socket is connected to the CLK input on the CDP1802. An external oscillator, therefore, may be connected to the CPU by connecting its output into the lower side of the socket.

When the XTAL switch is in the OUT position, the crystal or oscillator in the system under test is used.

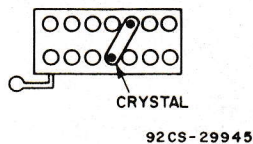


Fig. 39 — Crystal connection.

Specifications

Dimensions:

Length: 18½ inches

Width: 14½ inches

Height: 6 inches

Weight: 16 lbs. approx.

Controls:

Crystal In/Out

Reset

Power On/Off

Connectors:

CPU socket: 40 pin, zero insertion

Cable socket: 40 pin, zero insertion

Crystal socket: 14 pin, zero insertion

External memory connector: 44 pin edge connector; 0.156 inch pin spacing

Terminal input: 25-pin female Cinch connector

Terminal output: 25-pin male Cinch connector

External break input jack - dual banana

Memory disable jack - dual banana

Keyboard:

28 keys in 7 x 4 keyboard matrix; includes hex digit keys, function keys, and a shift key.

Display:

8-digit, 7-segment hexadecimal LED

Decimal points lighted as cursor

14 status-indicator LED's:

IDLE, MONITOR IN CONTROL, SC1, SC0, WAIT, CLEAR, Q, INTERRUPT, DMAIN, DMAOUT, EF1, EF2, EF3, and EF4

Power Requirements:

110/220 V ac, 50/60 Hz

Internal Power Supply:

Logic supply tracks system under test: 4 to 11 V at 500 mA

+5 V dc at 500 mA for LED's

+12 V dc at 200 mA for Terminal Interface

—12 V dc at 100 mA for Terminal Interface

Operating Temperature Range: 0 to 43°C

Cabling Supplied: AC power cord - 8 feet

System Cable: 40 wire, 3 feet long, terminated both ends in 40-pin Textool male connector

Terminal Interface: (input and output)
 20 mA or RS232C (EIA)
 100, 300, or 1200 baud

Memory:
 RAM: 256 bytes
 ROM: 6 kilobytes

System Clock:
 Uses clock from system under test to run user
 program

Internal clock: 2.112 MHz, crystal controlled

Monitor Loading on System Under Test:
 Power supply: 9.8 kilohms to ground
 Bus loading: 25 picofarads (typ)
 1 megohm (min)

Self-Test Card:
 Plug-in card for checking Micromonitor
 operation.

Example Session

To provide a more detailed introduction to use of the Micromonitor, a sample debugging session from a terminal follows. It is given for a Micromonitor/CDS II setup where the Micromonitor is being used to control operation of the COSMAC Development System (CDP18S805). This setup is accomplished by removing the CPU and plugging in the Micromonitor cable as previously described. The CPU is socketed on the CDSII and readily accessible for this purpose. As before, Micromonitor-generated output is underlined and additional comments are enclosed in parentheses. Arbitrary hexadecimal numbers are denoted by an X. The user will find it helpful to follow along on his own CDSII.

Consider the following program:

```

0000          0001 ORG #30
0030 F8FF    0002 LDI #FF .. #FF INTO D
0032 A4      0003 PLO R4 .. D INTO R4.0
0033 B4      0004 PHI R4 .. D INTO R4.1
0034 24      0005 DEC R4 .. DEC R4
0035 94      0006 GHI R4 .. R4.1 INTO D
0036 FC01    0007 ADI 1 .. ADD 1 TO D
0038 E2      0008 SEX R2 .. SET X TO R2
0039 3334    0009 BDF #34 .. DF, BRANCH BACK
003B 00      0010 IDL      .. ELSE, IDLE

```

The program is entered into the CDS memory with the following command.

```

_ IM30 F8FFA4B42494FC01E2333400 (CR)
_

```

Proper loading may be verified by examining memory

```

_ ?M30 C (CR)
0030 F8FF A4B4 2494 FC01 E233 3400
_

```

This sample program will be run with P=3.

```

_ !P3/
_ ?P/ 3
_

```

The current value of the 16 registers is now examined.

```

_ ?R (CR)
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
_

```

If the CDS has not executed a user program since the last RESET, RUN U sequence, the value in RC will be #80EF, the address of the delay routine used by UT20. This subject is explained in the Operator's Manual for the RCA COSMAC Development System II CDP18S005, MPM-216.

```

_ !R3 30/ (set R3 to point to program)
_ ?R3/ 0030
_ ?X/ x
_ $N:2/ (execute 2 instructions)
% (monitor not in control)
->
_ ?R3/ 0033 (program counter R3 has advanced 3 bytes)
_ ?R4/ xxFF (R4.0 has been loaded)
_ ?D/ FF
_ $$:2/ (step through the next instruction)
0 0033 B4 (fetch cycle: SC, address, data)
1 xxFF FF (execute cycle)
->

```



```

. ?R3/ 0034
. ?R4/ FFFF
. !BS R(39)/ (set a breakpoint at address 0039)
. $P/ (go to the breakpoint)
%
-> R(0039) (breakpoint reached)
. ?R3/ 0034 (note R3 points to next instruction)
. ?R4/ FFFE
. ?F/ 1
. $P:5/ (logs through the breakpoint 5 times)
%
-> R(0039) (break taken on read)
. ?R4/ FFF9
. !BS I/ (set a break on idle)
. !BC R/ (remove break on read)
. $P/ (continue)

```

```

%
-> I (break taken on idle)
. ?F/ 0 (DF now equals zero)
. ?R3/ 003B (program counter points at the idle instruction)
. ?R4/ FFFF
. ?L3/ (read the last 3 log entries)
P R(P) X D
3 003B 2 FF (present entry at idle break)
3 0034 2 00 (previous entries at read break)
3 0034 2 00

```

For the sake of clarity, many Micromonitor commands were not used in the above example. Their usage, however, is similar to that shown above. Readers are encouraged to experiment with similar simple programs to learn the Micromonitor commands before attempting to debug an actual system.

Appendix A - RCA Micromonitor Keyboard Command Summary

Function	Question Sequence	Display	Modify Sequence *	Function	Command Sequence	Display																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
Memory [▲]	M aaaa	aaaa, M(aaaa)	↔ Data Key(s) ENT (increments aaaa)	Run, stop after hhhh breaks [▲]	\$P aaaa ↔ hhhh ENT	NOTE: X=P=0 for specified aaaa. X,P are unchanged and aaaa=R(P) for unspecified aaaa.																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
Register [▲]	R n	R(n), n	↔ Data Key(s) ENT (increments n)	Run, do hhhh instructions [▲]	\$N aaaa ↔ hhhh ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
				Run, do hhhh cycles [▲]	\$S aaaa ↔ hhhh ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
DF [▲]	F-D	DF, D	Data Key(s) ENT	Enable Terminal [▲]	SH 10 30 120 LOG																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
DF [▲]	F-D	DF, D	↔ Data Key ENT	Data Log [▲]	SH ENT 	Logn, D, R(P) .. Logn, D,X,P																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
X [▲]	X-P	X, P	↔ Data Key ENT			(Data logged after break in \$P or instruction in \$N)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
X [▲]	X-P	X, P	Data Key ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
IE [▲]	SH IE-T	IE, T	↔ Data Key ENT	Input [▲]	SH IN p ENT	p, Data(p)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
T [▲]	SH IE-T	IE, T	Data Key(s) ENT	Output [▲]	SH OUT aaaa ↔ p ↔ Data ENT	Data, p, aaaa																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
Q [▲]	SH Q	Q	ENT (toggles)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
Wait	SH WAIT	Wait	ENT (toggles)	DMAI	SH DMAI	NOTE: Sets request. Reset is by RR or proper S2, S3 response.																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
Clear	SH CLR	Clear	ENT (toggles)	DMAO	SH DMAO																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
External Memory	SH EXM	External Memory	ENT (toggles)	Interrupt	SH INT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
Inhibit System Requests	SH IR	Inhibit Requests	ENT (toggles)	Reset Requests	SH RR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
External Flags	SH EFf	EFf	ENT (toggles)	Manual Break	BK																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
Break Conditions [▲]	BC	<table><tr><td>E</td><td>E</td><td>E</td><td>E</td><td>E</td><td>I</td><td>S</td><td>M</td></tr><tr><td>F</td><td>F</td><td>F</td><td>F</td><td>X</td><td>D</td><td>3</td><td>A</td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>T</td><td>L</td><td></td><td>S</td></tr></table>	E	E	E	E	E	I	S	M	F	F	F	F	X	D	3	A	1	2	3	4	T	L		S	↔ . . . Data Key ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
E	E	E	E	E	I	S	M																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
F	F	F	F	X	D	3	A																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
1	2	3	4	T	L		S																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							

Wave cursor with repeated ← depressions. Then, enter new value (0-9) and push ENT. For MAS, enter 0 (no stop), 1 (read), 2 (write), or 3 (either). Enabling MAS requires an address value entry.

* Modify Sequence must be preceded by Question Sequence.

▲ For these functions, the Micromonitor must be in control.

N ↔ Move cursor (decimal points) Prompt — All decimal points
O Shift Prompt — Alternate decimal points
T aaaa is an address n is a register number (0-F)
E f is a flag number (1-4) p is a port number (1-7)
S hhhh is a hex number

WARNING: If Micromonitor is OFF, do NOT apply power to the System Under Test.

Appendix B - RCA Micromonitor Terminal Command Summary

Function	Question	Modify	Defaults	Function	Command	Defaults
Memory [▲]	?Maaa hhhh	!Maaa xxxx	For ?M only aaaa=0000 hhhh=1	Run, stop after hhhh breaks [▲]	\$Paaa:hhhh	} aaaa=R(P) hhhh=1
Register [▲]	?Rn	!Rn hhhh	For ?Rn n=all For !Rn n=0 hhhh=0	Run, do hhhh instructions [▲]	\$Naaa:hhhh	
				Run, do hhhh machine cycles [▲]	\$Saaa:hhhh	
For above three commands, X,P are unchanged for unspecified aaaa; X=P=0 for specified aaaa. After completion —> is displayed followed by Break Condition (M=Manual)						
D [▲]	?D	!Dhh	hh=00	Return to Micromonitor Keyboard [▲]	\$K	
DF [▲]	?F	!Fb	b=0			
X [▲]	?X	!Xn	n=0			
P [▲]	?P	!Pn	n=0			
IE [▲]	?IE	!IEb	b=0			
T [▲]	?T	!Thh	hh=00			
Q [▲]	?Q	!Qb	b=0			
Wait	?W	!Wb	b=0			
Clear	?C	!Cb	b=0			
External Memory	?EM	!EMb	b=0			
Inhibit System Requests	?IR	!IRb	b=0			
External Flags	?EF	!EFf b	f=all b=0			
Data Log [▲]	?Lh	After break in \$P or instruc- tion in \$N.	h=all			
Break Conditions	?BS	!BSc [set] [▲] !BCc [clear] [▲]	c=all			
I/O Devices [▲]	?Ip	!Oaaaa:hh:p	aaaa=0 hh=M(aaaa) p=1			

WARNING: If Micromonitor is OFF, do NOT apply power to the System Under Test.

Note: On the plastic laminated Instruction Card MPM-921 supplied with Micromonitor CDP18S030, the command for DMAOUT in the right-hand column under RCA Micromonitor Terminal Commands is incomplete. It should read as shown above.

Appendix C -

RCA COSMAC Microprocessor CDP1802

Instruction Summary

The COSMAC instruction summary is given in Tables I and II. Hexadecimal notation is used to refer to the 4-bit binary codes.

In all registers bits are numbered from the least significant bit (LSB) to the most significant bit (MSB) starting with 0.

R(W): Register designated by W, where W=N or X, or P

R(W).0: Lower-order byte of R(W)

R(W).1: Higher-order byte of R(W)

NO = Least significant Bit of N Register

Operation Notation

$M(R(N)) \rightarrow D; R(N) + 1$

This notation means: The memory byte pointed to by R(N) is loaded into D, and R(N) is incremented by 1.

TABLE I — INSTRUCTION SUMMARY
by Class of Operation

Register Operations

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
INCREMENT REG N	INC	1N	$R(N) + 1$
DECREMENT REG N	DEC	2N	$R(N) - 1$
INCREMENT REG X	IRX	60	$R(X) + 1$
GET LOW REG N	GLO	8N	$R(N).0 \rightarrow D$
PUT LOW REG N	PLO	AN	$D \rightarrow R(N).0$
GET HIGH REG N	GHI	9N	$R(N).1 \rightarrow D$
PUT HIGH REG N	PHI	BN	$D \rightarrow R(N).1$

Memory Reference

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
LOAD VIA N	LDN	0N	$M(R(N)) \rightarrow D; \text{FOR } N \text{ NOT } 0$
LOAD ADVANCE	LDA	4N	$M(R(N)) \rightarrow D; R(N) + 1$
LOAD VIA X	LDX	F0	$M(R(X)) \rightarrow D$
LOAD VIA X AND ADVANCE	LDXA	72	$M(R(X)) \rightarrow D; R(X) + 1$
LOAD IMMEDIATE	LDI	F8	$M(R(P)) \rightarrow D; R(P) + 1$
STORE VIA N	STR	5N	$D \rightarrow M(R(N))$
STORE VIA X AND DECREMENT	STXD	73	$D \rightarrow M(R(X)); R(X) - 1$

Logic Operations♦♦

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
OR	OR	F1	$M(R(X)) \text{ OR } D \rightarrow D$
OR IMMEDIATE	ORI	F9	$M(R(P)) \text{ OR } D \rightarrow D; R(P) + 1$
EXCLUSIVE OR	XOR	F3	$M(R(X)) \text{ XOR } D \rightarrow D$
EXCLUSIVE OR IMMEDIATE	XRI	FB	$M(R(P)) \text{ XOR } D \rightarrow D; R(P) + 1$
AND	AND	F2	$M(R(X)) \text{ AND } D \rightarrow D$
AND IMMEDIATE	ANI	FA	$M(R(P)) \text{ AND } D \rightarrow D; R(P) + 1$
SHIFT RIGHT	SHR	F6	SHIFT D RIGHT, $LSB(D) \rightarrow DF$, $0 \rightarrow MSB(D)$
SHIFT RIGHT WITH CARRY	SHRC	76♦	SHIFT D RIGHT, $LSB(D) \rightarrow DF$, $DF \rightarrow MSB(D)$
RING SHIFT RIGHT	RSHR		
SHIFT LEFT	SHL	FE	SHIFT D LEFT, $MSB(D) \rightarrow DF$, $0 \rightarrow LSB(D)$
SHIFT LEFT WITH CARRY	SHLC	7E♦	SHIFT D LEFT, $MSB(D) \rightarrow DF$, $DF \rightarrow LSB(D)$
RING SHIFT LEFT	RSHL		

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.

♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF.

Arithmetic Operations♦♦

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
ADD	ADD	F4	$M(R(X)) + D \rightarrow DF, D$
ADD IMMEDIATE	ADI	FC	$M(R(P)) + D \rightarrow DF, D; R(P) + 1$
ADD WITH CARRY	ADC	74	$M(R(X)) + D + DF \rightarrow DF, D$
ADD WITH CARRY, IMMEDIATE	ADCI	7C	$M(R(P)) + D + DF \rightarrow DF, D; R(P) + 1$
SUBTRACT D	SD	F5	$M(R(X)) - D \rightarrow DF, D$
SUBTRACT D IMMEDIATE	SDI	FD	$M(R(P)) - D \rightarrow DF, D; R(P) + 1$
SUBTRACT D WITH BORROW	SDB	75	$M(R(X)) - D - (NOT DF) \rightarrow DF, D$
SUBTRACT D WITH BORROW, IMMEDIATE	SDBI	7D	$M(R(P)) - D - (NOT DF) \rightarrow DF, D; R(P) + 1$
SUBTRACT MEMORY	SM	F7	$D - M(R(X)) \rightarrow DF, D$
SUBTRACT MEMORY IMMEDIATE	SMI	FF	$D - M(R(P)) \rightarrow DF, D; R(P) + 1$
SUBTRACT MEMORY WITH BORROW	SMB	77	$D - M(R(X)) - (NOT DF) \rightarrow DF, D$
SUBTRACT MEMORY WITH BORROW, IMMEDIATE	SMBI	7F	$D - M(R(P)) - (NOT DF) \rightarrow DF, D; R(P) + 1$

Branch Instructions — Short Branch

SHORT BRANCH	BR	30	$M(R(P)) \rightarrow R(P).0$
NO SHORT BRANCH (SEE SKP)	NBR	38♦	$R(P) + 1$
SHORT BRANCH IF D=0	BZ	32	IF D=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF D NOT 0	BNZ	3A	IF D NOT 0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF DF=1	BDF	33♦	IF DF=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF POS OR ZERO	BPZ		
SHORT BRANCH IF EQUAL OR GREATER	BGE		
SHORT BRANCH IF DF=0	BNF	3B♦	IF DF=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF MINUS	BM		
SHORT BRANCH IF LESS	BL		
SHORT BRANCH IF Q=1	BQ	31	IF Q=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF Q=0	BNQ	39	IF Q=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF1=1 (1 = VSS)	B1	34	IF EF1=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF1=0 (0 = VCC)	BN1	3C	IF EF1=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF2=1 (1 = VSS)	B2	35	IF EF2=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF2=0 (0 = VCC)	BN2	3D	IF EF2=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF3=1 (1 = VSS)	B3	36	IF EF3=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF3=0 (0 = VCC)	BN3	3E	IF EF3=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF4=1 (1 = VSS)	B4	37	IF EF4=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF4=0 (0 = VCC)	BN4	3F	IF EF4=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$

Branch Instructions — Long Branch

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
LONG BRANCH	LBR	C0	$M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ $R(P) + 2$
NO LONG BRANCH (SEE LSKP)	NLBR	C8♦	
LONG BRANCH IF D=0	LBZ	C2	IF D=0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF D NOT 0	LBNZ	CA	IF D NOT 0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF DF=1	LBDF	C3	IF DF=1, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF DF=0	LBNF	CB	IF DF=0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF Q=1	LBQ	C1	IF Q=1, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF Q=0	LBNQ	C9	IF Q=0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$

Skip Instructions

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
SHORT SKIP (SEE NBR)	SKP	38♦	$R(P) + 1$
LONG SKIP (SEE NLBR)	LSKP	C8♦	$R(P) + 2$
LONG SKIP IF D=0	LSZ	CE	IF D=0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF D NOT 0	LSNZ	C6	IF D NOT 0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF DF=1	LSDF	CF	IF DF=1, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF DF=0	LSNF	C7	IF DF=0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF Q=1	LSQ	CD	IF Q=1, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF Q=0	LSNQ	C5	IF Q=0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF IE=1	LSIE	CC	IF IE=1, $R(P) + 2$ ELSE CONTINUE

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.

♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF.

Control Instructions

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
IDLE	IDL	00	WAIT FOR DMA OR INTERRUPT; M(R(0))→BUS
NO OPERATION	NOP	C4	CONTINUE
SET P	SEP	DN	N→P
SET X	SEX	EN	N→X
SET Q	SEQ	7B	1→Q
RESET Q	REQ	7A	0→Q
SAVE	SAV	78	T→M(R(X))
PUSH X,P TO STACK	MARK	79	(X,P)→T; (X,P)→M(R(2))
			THEN P→X; R(2)-1
RETURN	RET	70	M(R(X))→(X,P); R(X) +1
			1→IE
DISABLE	DIS	71	M(R(X))→(X,P); R(X) +1
			0→IE

Input-Output Byte Transfer

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
OUTPUT 1	OUT 1	61	M(R(X))→BUS; R(X) +1; N LINES = 1
OUTPUT 2	OUT 2	62	M(R(X))→BUS; R(X) +1; N LINES = 2
OUTPUT 3	OUT 3	63	M(R(X))→BUS; R(X) +1; N LINES = 3
OUTPUT 4	OUT 4	64	M(R(X))→BUS; R(X) +1; N LINES = 4
OUTPUT 5	OUT 5	65	M(R(X))→BUS; R(X) +1; N LINES = 5
OUTPUT 6	OUT 6	66	M(R(X))→BUS; R(X) +1; N LINES = 6
OUTPUT 7	OUT 7	67	M(R(X))→BUS; R(X) +1; N LINES = 7
INPUT 1	INP 1	69	BUS→M(R(X)); BUS→D; N LINES = 1
INPUT 2	INP 2	6A	BUS→M(R(X)); BUS→D; N LINES = 2
INPUT 3	INP 3	6B	BUS→M(R(X)); BUS→D; N LINES = 3
INPUT 4	INP 4	6C	BUS→M(R(X)); BUS→D; N LINES = 4
INPUT 5	INP 5	6D	BUS→M(R(X)); BUS→D; N LINES = 5
INPUT 6	INP 6	6E	BUS→M(R(X)); BUS→D; N LINES = 6
INPUT 7	INP 7	6F	BUS→M(R(X)); BUS→D; N LINES = 7

- ◆NOTE THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.
- ◆◆NOTE THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF

TABLE II - INSTRUCTION SUMMARY
By Numerical Order

OPERATION CODE	OPERAND	MNEMONIC	NAME	MACHINE CYCLE	NUMBER OF PROGRAM BYTES
00	—	IDL	IDLE	2	1
0N	REG N	LDN	LOAD VIA N	2	1
1N	REG N	INC	INCREMENT REG N	2	1
2N	REG N	DEC	DECREMENT REG N	2	1
30	ADDRESS	BR	SHORT BRANCH	2	2
31	ADDRESS	BQ	SHORT BRANCH IF Q=1	2	2
32	ADDRESS	BZ	SHORT BRANCH IF D=0	2	2
33	ADDRESS	BDF	SHORT BRANCH IF DF=1	2	2
—	ADDRESS	BPZ	SHORT BRANCH IF POS OR ZERO	2	2
—	ADDRESS	BGE	SHORT BRANCH IF EQUAL OR GREATER	2	2
34	ADDRESS	B1	SHORT BRANCH IF EF1=1	2	2
35	ADDRESS	B2	SHORT BRANCH IF EF2=1	2	2
36	ADDRESS	B3	SHORT BRANCH IF EF3=1	2	2
37	ADDRESS	B4	SHORT BRANCH IF EF4=1	2	2
38	ADDRESS	NBR	NO SHORT BRANCH	2	2
—	ADDRESS	SKP	SHORT SKIP	2	1
39	ADDRESS	BNQ	SHORT BRANCH IF Q=0	2	2
3A	ADDRESS	BNZ	SHORT BRANCH IF D NOT 0	2	2
3B	ADDRESS	BNF	SHORT BRANCH IF DF=0	2	2
—	ADDRESS	BM	SHORT BRANCH IF MINUS	2	2
—	ADDRESS	BL	SHORT BRANCH IF LESS	2	2

INSTRUCTION SUMMARY (CONT'D)

OPERATION CODE	OPERAND	MNEMONIC	NAME	MACHINE CYCLES	NUMBER OF PROGRAM BYTES
3C	ADDRESS	BN1	SHORT BRANCH IF EF1=0	2	2
3D	ADDRESS	BN2	SHORT BRANCH IF EF2=0	2	2
3E	ADDRESS	BN3	SHORT BRANCH IF EF3=0	2	2
3F	ADDRESS	BN4	SHORT BRANCH IF EF4=0	2	2
4N	REG N	LDA	LOAD ADVANCE	2	1
5N	REG N	STR	STORE VIA N	2	1
60	—	IRX	INCREMENT REG X	2	1
61	DEVICE 1	OUT1	OUTPUT1	2	1
62	DEVICE 2	OUT2	OUTPUT2	2	1
63	DEVICE 3	OUT3	OUTPUT3	2	1
64	DEVICE 4	OUT4	OUTPUT4	2	1
65	DEVICE 5	OUT5	OUTPUT5	2	1
66	DEVICE 6	OUT6	OUTPUT6	2	1
67	DEVICE 7	OUT7	OUTPUT7	2	1
68			DO NOT USE		
69	DEVICE 1	INP1	INPUT1	2	1
6A	DEVICE 2	INP2	INPUT2	2	1
6B	DEVICE 3	INP3	INPUT3	2	1
6C	DEVICE 4	INP4	INPUT4	2	1
6D	DEVICE 5	INP5	INPUT5	2	1
6E	DEVICE 6	INP6	INPUT6	2	1
6F	DEVICE 7	INP7	INPUT7	2	1
70	—	RET	RETURN	2	1
71	—	DIS	DISABLE	2	1
72	—	LDXA	LOAD VIA X, ADVANCE	2	1
73	—	STXD	STORE VIA X AND DECREMENT	2	1
74	—	ADC	ADD WITH CARRY	2	1
75	—	SDB	SUBTRACT D WITH BORROW	2	1
76 } — }	—	SHRC	SHIFT RIGHT WITH CARRY	2	1
	—	RSHR	RING SHIFT RIGHT	2	1
77	—	SMB	SUBTRACT MEMORY WITH BORROW	2	1
78	—	SAV	SAVE	2	1
79	—	MARK	PUSH X,P TO STACK	2	1
7A	—	REQ	RESET Q	2	1
7B	—	SEQ	SET Q	2	1
7C	DATA	ADCI	ADD WITH CARRY IMMEDIATE	2	2

INSTRUCTION SUMMARY (CONT'D)

OPERATION CODE	OPERAND	MNEMONIC	NAME	MACHINE CYCLES	NUMBER OF PROGRAM BYTES
7D	DATA	SDBI	SUBTRACT D WITH BORROW IMMEDIATE	2	2
7E	—	SHLC	SHIFT LEFT WITH CARRY	2	1
	—	RSHL	RING SHIFT LEFT	2	1
7F	DATA	SMBI	SUBTRACT MEMORY WITH BORROW, IMMEDIATE	2	2
8N	REG N	GLO	GET LOW REG N	2	1
9N	REG N	GHI	GET HIGH REG N	2	1
AN	REG N	PLO	PUT LOW REG N	2	1
BN	REG N	PHI	PUT HIGH REG N	2	1
C0	ADDRESS	LBR	LONG BRANCH	3	3
C1	ADDRESS	LBQ	LONG BRANCH IF Q=1	3	3
C2	ADDRESS	LBZ	LONG BRANCH IF D=0	3	3
C3	ADDRESS	LBDF	LONG BRANCH IF DF=1	3	3
C4	—	NOP	NO OPERATION	3	1
C5	—	LSNQ	LONG SKIP IF Q=0	3	1
C6	—	LSNZ	LONG SKIP IF D NOT 0	3	1
C7	—	LSNF	LONG SKIP IF DF=0	3	1
C8 } — }	— ADDRESS	LSKP NLBR	LONG SKIP NO LONG BRANCH	3 3	1 3
C9	ADDRESS	LBNO	LONG BRANCH IF Q=0	3	3
CA	ADDRESS	LBNZ	LONG BRANCH IF D NOT 0	3	3
CB	ADDRESS	LBNF	LONG BRANCH IF DF=0	3	3
CC	—	LSIE	LONG SKIP IF IE=1	3	1
CD	—	LSQ	LONG SKIP IF Q=1	3	1
CE	—	LSZ	LONG SKIP IF D=0	3	1
CF	—	LSDF	LONG SKIP IF DF=1	3	1
DN	REG N	SEP	SET P	2	1
EN	REG N	SEX	SET X	2	1
F0	—	LDX	LOAD VIA X	2	1
F1	—	OR	OR	2	1
F2	—	AND	AND	2	1

INSTRUCTION SUMMARY (CONT'D)

OPERATION CODE	OPERAND	MNEMONIC	NAME	MACHINE CYCLES	NUMBER OF PROGRAM BYTES
F3	—	XOR	EXCLUSIVE OR	2	1
F4	—	ADD	ADD	2	1
F5	—	SD	SUBTRACT D	2	1
F6	—	SHR	SHIFT RIGHT	2	1
F7	—	SM	SUBTRACT MEMORY	2	1
F8	DATA	LDI	LOAD IMMEDIATE	2	2
F9	DATA	ORI	OR IMMEDIATE	2	2
FA	DATA	ANI	AND IMMEDIATE	2	2
FB	DATA	XRI	EXCLUSIVE OR IMMEDIATE	2	2
FC	DATA	ADI	ADD IMMEDIATE	2	2
FD	DATA	SDI	SUBTRACT D IMMEDIATE	2	2
FE	—	SHL	SHIFT LEFT	2	1
FF	DATA	SMI	SUBTRACT MEMORY IMMEDIATE	2	2

Interpretation of DF

	DF	Carry Generated	Borrow Generated	D
After Addition	1	Yes		
	0	No		
After Subtraction	1		No	Positive Number
	0		Yes	Negative Number 2's complement

Hexadecimal Code

HEX	BINARY	HEX	BINARY
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

COSMAC Register Summary

D	8 Bits	Data Register (Accumulator)	N	4 Bits	Holds Low-Order Instr. Digit
DF	1 Bit	Data Flag (ALU Carry)	I	4 Bits	Holds High-Order Instr. Digit
R	16 Bits	1 of 16 Scratchpad Registers	T	8 Bits	Holds old X, P after Interrupt (X is high byte)
P	4 Bits	Designates which register is Program Counter	IE	1 Bit	Interrupt Enable Flip Flop
X	4 Bits	Designates which register is Data Pointer	Q	1 Bit	Output Flip Flop

Appendix D - Logic Diagrams for RCA COSMAC Micromonitor CDP18S030

Fig. No.	Page No.
D1 – Microprocessor and Memory Section	50
D2 – User Microprocessor and Cable Sockets	51
D3 – Control Logic	52
D4 – Interface and I/O Decode	53
D5 – Breakpoint Logic	54
D6 – Keyboard and Display Logic	55
D7 – Terminal Interface	56
D8 – Capacitor Connections	57
D9 – Unused Gates	57
D10 – Layout Diagram	58
Parts List for RCA COSMAC Micromonitor CDP18S030	59



Fig. D2 — User microprocessor and cable socket.

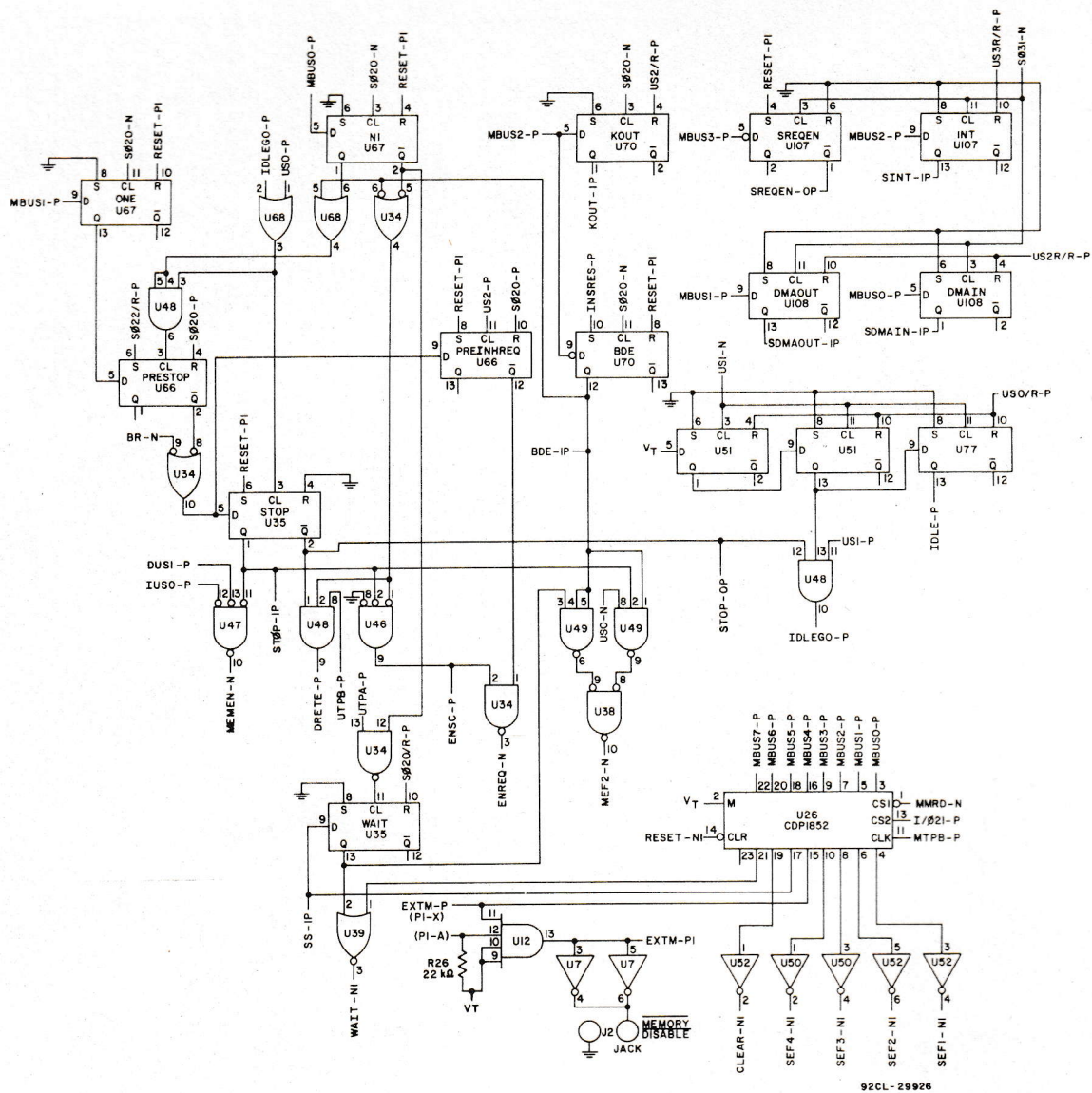


Fig. D3 - Control logic.



Fig. D4 – Interface and I/O decode.

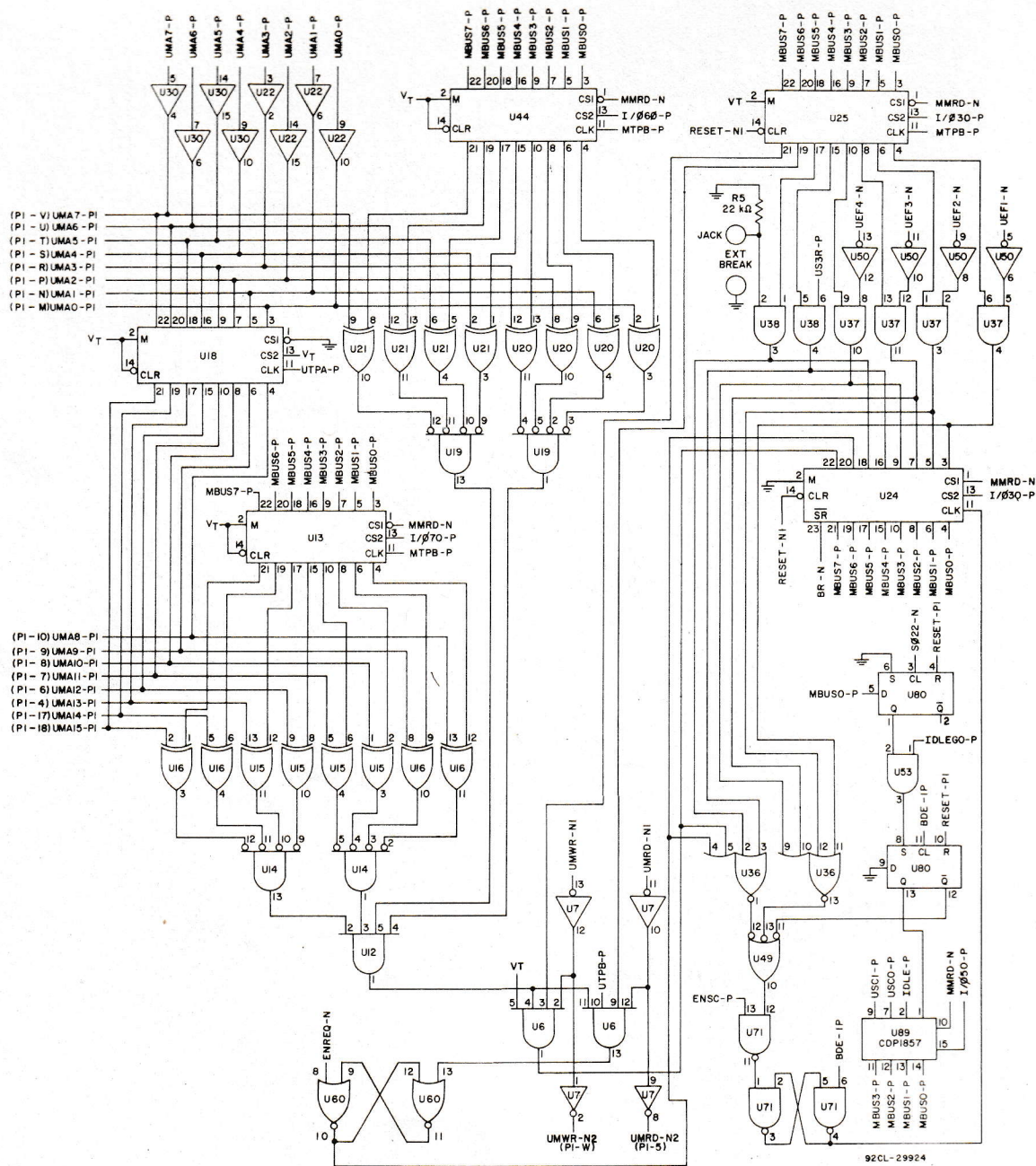


Fig. D5 — Breakpoint logic.



Fig. D6 – Keyboard and display logic.

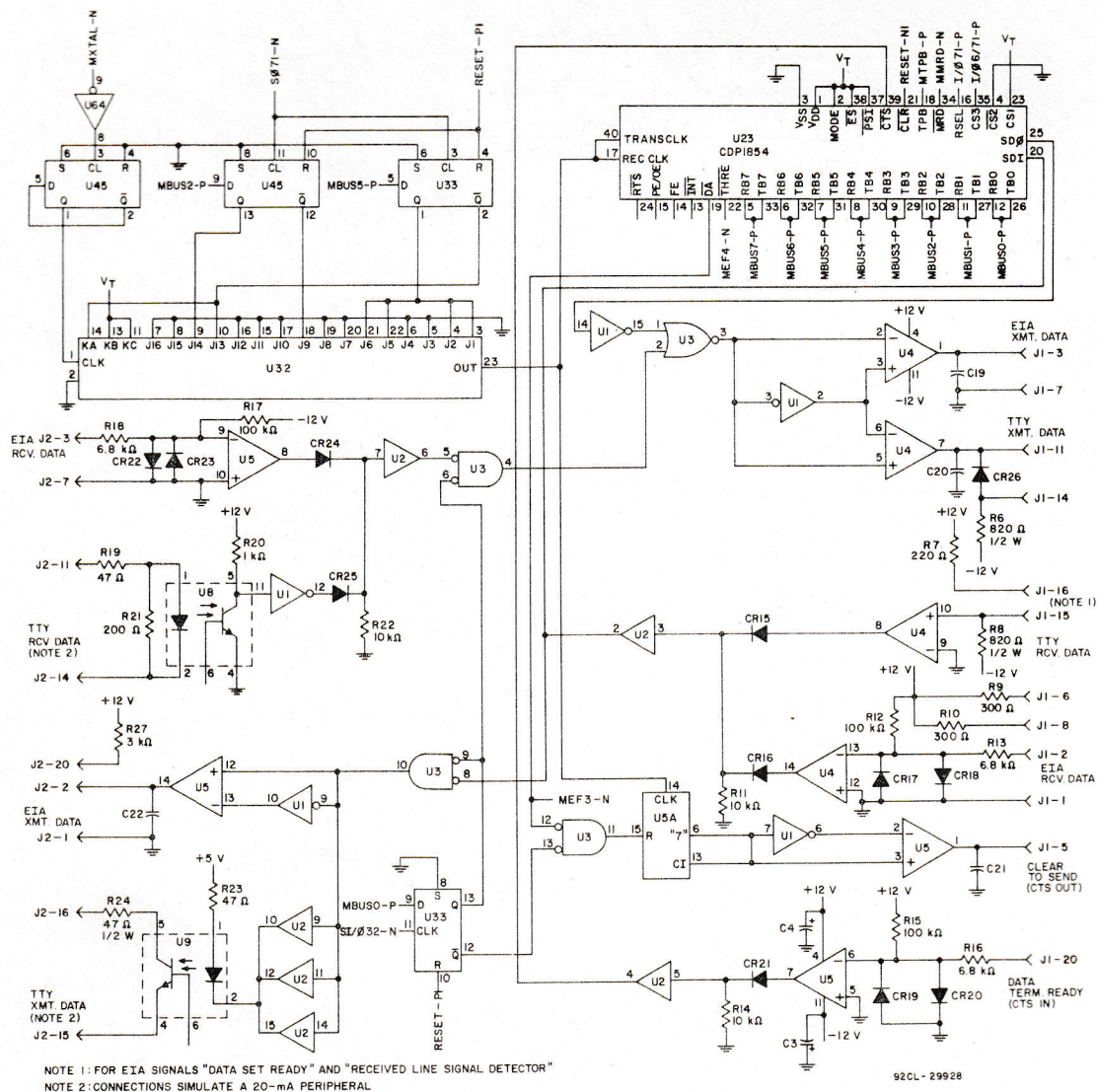


Fig. D7 - Terminal interface.

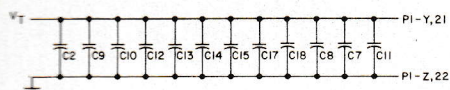


Fig. D8 – Capacitor connections.

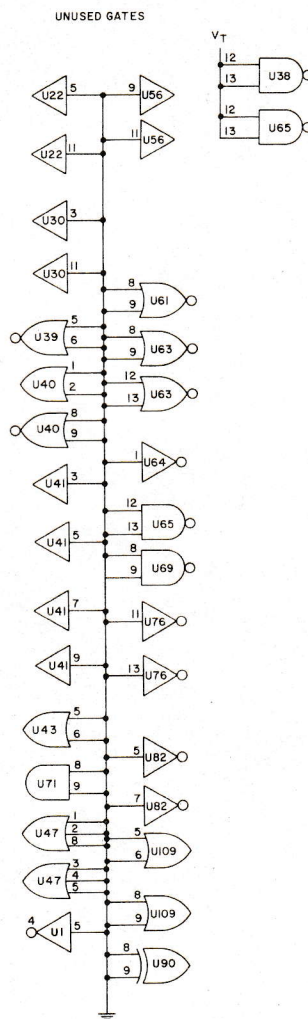


Fig. D9 – Unused gates.

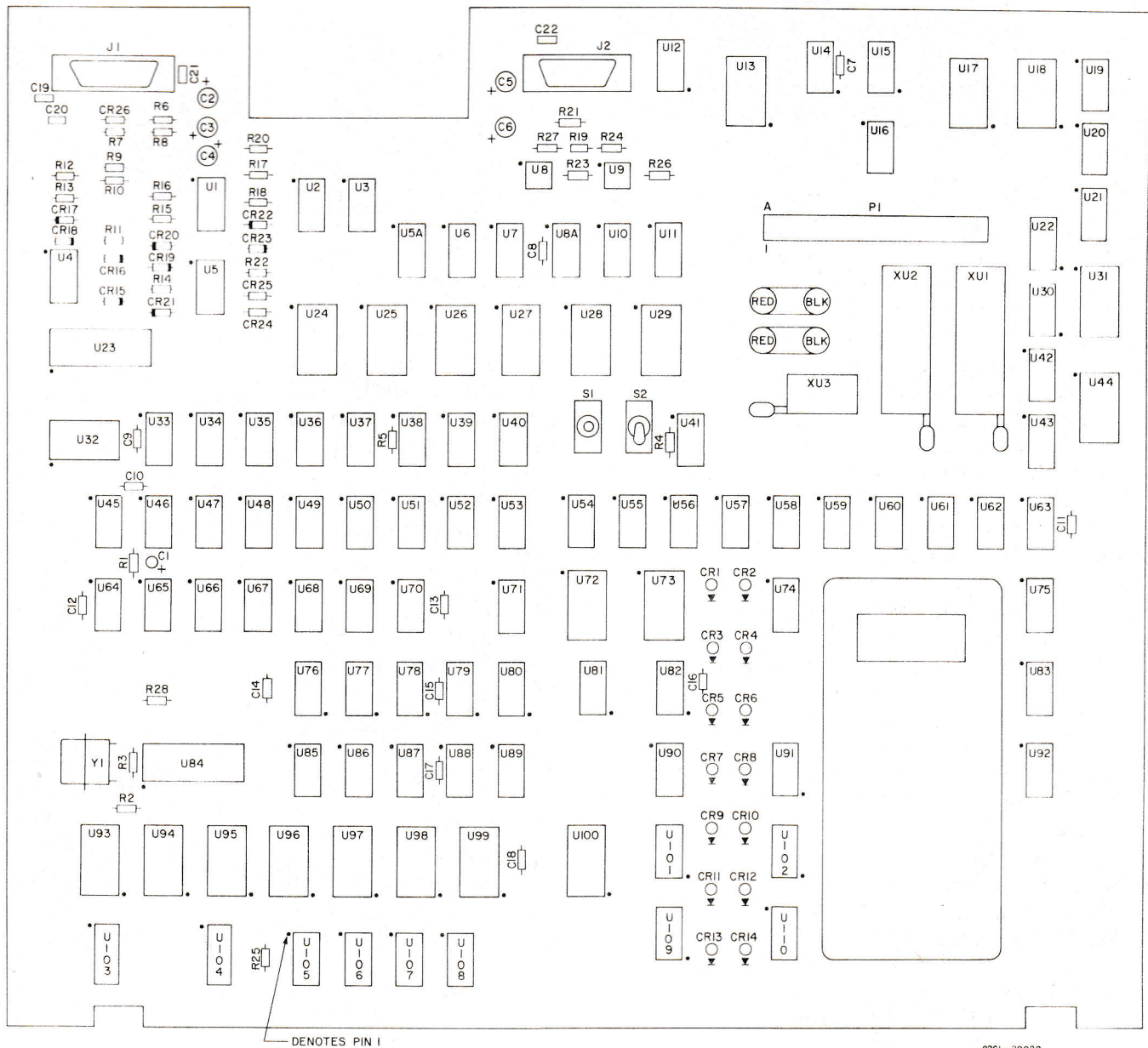


Fig. D10 — Layout diagram.

Parts List for RCA COSMAC Micromonitor CDP18S030
(See Figs. D1 through D10)

C1=1.5 μ F, 25 V
C2-C6=10 μ F, 25 V
C7-C18=0.1 μ F, 20 V
C19-C22=200 pF, 1000 V
CR1-CR14=LED
CR15-CR26=1N914
J1=connector, 25-pin female
J2=connector, 25-pin male
R1, R4, R12, R15, R17=100 kilohms, 0.25 W
R2, R5, R25, R26, R28=22 kilohms, 0.25 W
R3=10 megohms, 0.25 W
R6, R8=820 ohms, 0.5 W
R7=220 ohms, 0.5 W
R9, R10=300 ohms, 0.25 W
R11, R14, R22=10 kilohms, 0.25 W
R13, R16, R18=6.8 kilohms, 0.25 W
R19, R23=47 ohms, 0.25 W
R20=1 kilohm, 0.25 W
R21=200 ohms, 0.25 W
R24=47 ohms, 0.5 W
R27=3 kilohms, 0.25 W
S1=SPST
S2=SPDT
U1, U56=CD4049UB/A
U2, U22, U30, U41, U57, U58, U81, U82=CD4050B/A
U3, U39, U60, U62, U63=CD4001UB
U4, U5=CA324
U5A=CD4017B/A
U6, U12=CD4082B
U7, U50, U52, U64, U76=CD4069UB
U8, U9=Opto-Isolator 2460147 (Monsanto)
U8A, U91, U92, U106=resistor network, 22 kilohms
U10, U11=CDP1856D
U13, U17, U18, U24-U29, U31, U44, U72, U73,
U100=CDP1852D
U14, U19, U36=CD4002UB
U15, U16, U20, U21, U90, U101=CD4070B
U23=CDP1854D
U32=CD4059A
U33, U35, U45, U51, U66, U67, U70, U77, U78, U80,
U107, U108=CD4013B
U34, U69, U71=CD4011UB
U37, U38, U42, U53, U59, U86, U87, U88=CD4081B
U40, U43, U54, U61, U68, U109=CD4071B
U46=CD4025UB
U47=CD4075B
U48, U79=CD4073B
U49=CD4023UB
U55, U85=CDP1853D
U65=CD4093B
U74=resistor network, 470 ohms
U75=CA3081
U83, U102=CA3096
U84=CDP1802D
U89=CDP1857D
U93-U99=CDP1833D
U103, U104=CDP1822D
U105=CD4072B
U110=CD4066B/A
XU1, XU2=40-terminal socket
XU3=14-terminal socket
Y1=crystal, 2.112 MHz, JAN HT-6

Appendix E - Connector Pin Lists

External Memory Interface Connector Pin List

1	TPA	A	External Memory Deselect—N
2		B	5 Volts
3	TPB	C	BUS 0
4	MA13	D	BUS 1
5	MRD—N	E	BUS 2
6	MA12	F	BUS 3
7	MA11	H	BUS 4
8	MA10	J	BUS 5
9	MA9	K	BUS 6
10	MA8	L	BUS 7
11		M	MA0
12		N	MA1
13		P	MA2
14		R	MA3
15		S	MA4
16		T	MA5
17	MA14	U	MA6
18	MA15	V	MA7
19		W	MWR—N
20		X	External Memory Select—P
21	V _{DD} (400 mA)	Y	V _{DD}
22	GND	Z	GND

Terminal Input Connector (J1) Pin List

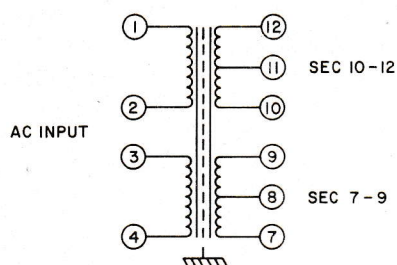
1	Ground
2	EIA from terminal
3	EIA to terminal
4	
5	Terminal clear to send
6	
7	Ground
8	
9	
10	
11	20 mA to printer (+)
12	
13	
14	20 mA to printer (—)
15	20 mA to keyboard (—)
16	20 mA to keyboard (+)
17	
18	
19	
20	Ready from terminal
21	
22	
23	
24	
25	

Terminal Output Connector (J2) Pin List

1	Ground
2	EIA Output to system
3	EIA Input from system*
4	
5	
6	
7	Ground
8	
9	
10	
11	20 mA to printer (+)
12	
13	
14	20 mA to printer (—)
15	20 mA to keyboard (—)
16	20 mA to keyboard (+)
17	
18	
19	
20	Clear to Send*
21	
22	
23	
24	
25	

*When 20-mA terminal is connected to J2, pins 3 and 20 must be connected together.

Appendix F - Transformer Connections for 115- or 230-Volt AC Operation



NOTE: PARALLEL WINDINGS 1-2 AND 3-4
FOR 115-VAC OPERATION
SERIES WINDINGS 1-2 AND 3-4
FOR 230-VAC OPERATION

92CS-29936

Appendix G -

Instructions for Converting a Model 33 Teletype Terminal from Half- to Full-Duplex Operation and from 60-mA to 20-mA Operation

For a Teletype* terminal connected for half-duplex operation, the following modifications can be made to convert it to full-duplex operation.

1. Locate the black terminal strip in the back. See Fig. G1.
2. Move the brown/yellow and white/blue wires from pins 3 or 4 to pin 5.

For Teletype terminals, connected for 60-mA operation, the following modifications can be made for 20-mA operation.

1. Move the violet wire from pin 8 to pin 9.
2. Move the blue wire connected to the current source resistor (a flat green resistor with four tabs located to the right of the keyboard) from the 750-ohm tab to the 1450-ohm tab.

*Registered Trademark, Teletype Corporation.

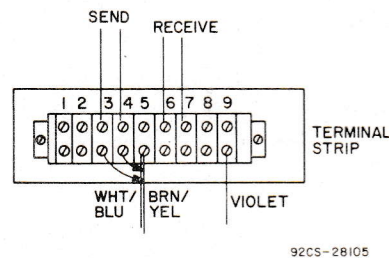
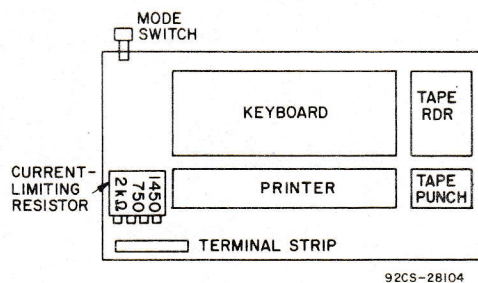


Fig.G1 — Location of and connections to terminal strip for Model 33 Teletype data terminal showing connections for 20-mA full-duplex operation.

Appendix H - Data Terminal-Micromonitor Connection Details

A. Check List for 20-mA Current Loop Interface (See Fig. 2, page 7)

1. Make sure that the TTY is configured for 20-mA current loop operation (See Appendix G).

2. Locate the four 20-mA current loop interface terminals on either the Jack J2 or the terminal strip TS of the TTY and make the connections to the Micromonitor J1 as shown in Table H-I.

TTY 33ASR		Micromonitor
TS	J2	
7	8	J1-11
6	7	J1-14
4	6	J1-15
3	5	J1-16

Table H-I - TTY - Micromonitor Connections

B. Check List for EIA RS232C Interface (See Fig. 2, page 7)

1. Consult the data terminal instruction manual for information on where to access the RS232C signals.

2. Connect the EIA RS232C interface signals from the data terminal to the Micromonitor as shown in Table H-II.

EIA RS232C

Pin Number	Signal Function	Micromonitor
1	Protective Ground	J1-1
2	Transmitted Data	J1-2
3	Received Data	J1-3
4	Request to Send	*
5	Clear to Send	J1-5
6	Data Set Ready	J1-6
7	Signal Ground	J1-7
8	Received Line Signal Detector	J1-8
20	Data Terminal Ready	J1-20

*No Connection Necessary

Table H-II - EIA RS232C data terminal - Micromonitor connections.

C. Check List for Data Terminal Shared by Micromonitor and CDS (See Fig. 3, page 7)

A data terminal may be used to control both a Micromonitor and a COSMAC Development System by connecting the data terminal to J1 of the Micromonitor (See Table H-I or H-II) and the CDS to J2. A possible application for this configuration would be in downloading a program from the CDS onto paper tape. The Micromonitor could then be used to load its external memory or the memory of the system under test from the paper tape.

1. When the CDS is to be controlled from the data terminal, make sure that the Micromonitor is in the Keyboard mode (i.e., the keyboard is in control of the Micromonitor).

2. Connect the CDS Terminal Interface module CDP18S507 and the Micromonitor as shown in Table H-III or H-IV.

CDS	Signal Function	Micromonitor
J2-1, 10	Ground	J2-1, 7
J2-3	CDS Serial Data Out	J2-3
J2-2	CDS Serial Data In	J2-2

Table H-III - CDS-Micromonitor connections for use with data terminal having EIA RS232C interface.

CDS	Signal Function	Micromonitor
J1-7	CDS Data	J2-11
J1-3	Out	J2-14
J1-4	CDS Data	J2-16
J1-8	In	J2-15

Table H-IV - CDS-Micromonitor connections for use with data terminal having 20-mA current loop interface.

D. Check List for Data Terminal Shared by Micromonitor and System Under Test (See Fig. 4, page 7).

1. When the System under test is to be controlled from the data terminal, make sure that the Micromonitor is in the Keyboard mode.

2. Connect the data terminal to the Micromonitor as shown in Table H-I or H-II (EIA or TTY).

3. Refer to the terminal interface instructions for the system under test for proper connections to the data terminal (EIA or TTY).

Appendix I - COSMAC Micromonitor Operating System (MOPS) CDP18S831

The Micromonitor Operating System (MOPS) is a software package developed to enhance the capabilities of the RCA CDP18S030 Micromonitor. The Micromonitor is a self-contained, powerful debugging tool for use with any system based on the CDP1802 COSMAC Microprocessor. It permits in-circuit debugging in real time so that both hardware and software problems can be efficiently identified. The Micromonitor Operating System (MOPS) CDP18S831 enhances Micromonitor performance by providing user access to the processing and storage capabilities of the COSMAC Development System CDP18S005 equipped with the Floppy Disk System CDP18S805.

The Micromonitor Operating System CDP18S831 includes a MOPS Diskette CDP18S830, a UART Module CDP18S508, and a connecting cable CDP18S511.

System Functions

The Micromonitor Operating System CDP18S831 provides an extended Micromonitor-type command set with commands of the following types:

1. Commands that allow the user to conveniently switch Micromonitor commands and responses to and from a variety of system peripherals.
2. Single commands that allow a more complete interrogation of the CPU state.
3. Commands for saving the system-under-test memory, registers, etc. in a disk file or for loading the system-under-test from a disk file.
4. Commands that allow a degree of automation in system debugging and testing.

With MOPS, the debugging techniques available to the user range from simple terminal-Micromonitor dialog to fully automated hands-off system testing with commands coming from disk files.

Appendix J - Operation of a Typical Developmental System

This Appendix discusses a typical developmental system configuration and the operating steps involved in editing, assembling, and debugging a program on a system composed of a COSMAC Development System II, a Floppy Disk System, a Micromonitor, and a data terminal. Fig. J1 shows such a system. Note that the terminal is connected to the Micromonitor and also, via the feedthrough feature, to the CDS. The terminal can thereby be used to communicate with either device. Because the object of the system configuration is to develop and debug a program on the CDS for possible later use in another environment, it is assumed that any necessary I/O hardware for the application under development is contained on a module plugged into the CDS. For a discussion of disabling two-level I/O's in CDS II, refer to "Two-Level I/O" in the Section **Hardware Structure of the CDS in the Operator Manual for the RCA CDS II CDP18S005, MPM-216.**

For the physical setup of Fig. J1, the following steps should be taken:

1. Connect the data terminal to the Micromonitor as shown in Table H-I or H-II of Appendix H.
2. Connect the Micromonitor to the CDS Terminal Interface Module (slot 14) as shown in Table H-III or H-IV of Appendix H.
3. Connect the Floppy Disk System cable to the CDS Disk Interface Module (slot 24).
4. Remove the CPU Module (slot 12) from the CDS and note the position of pin 1 of the CDP1802. Carefully remove the CDP1802 and install it into the CPU socket of the Micromonitor.

5. Install the cable between the Micromonitor and the CPU Module. Be sure to observe the proper cable and socket polarity. Plug the CPU Module back into the CDS.

6. Install a 2.0000-MHz crystal in the crystal socket of the Micromonitor and set the selector switch to IN. Only a crystal of 2.000 MHz exactly should be used for running RCA-supplied software.

7. Make any connections required to the External Break Input, Memory Disable Output, or insert an external memory module at this point.

Power may be supplied simultaneously to all systems if, for example, they are all plugged into a common power outlet strip. If not, they should be sequenced in the following order: Micromonitor, Data Terminal, CDS, and Floppy Disk System. To turn off power, reverse the sequence.

The following steps outline the procedure for creating a file, assembling it, loading the program into the CDS RAM, and, finally, debugging it using the Micromonitor.

8. Press the RESET switch on the Micromonitor. The decimal point prompt should appear and the MON light should come on.
9. Push RESET on the CDS.
10. Press the \$P and then press the ENT button on the Micromonitor to release Micromonitor control. The MON light should go out.
11. Press RUN U on the CDS, and press Carriage Return on the terminal to start UT20. An asterisk should appear on the terminal.

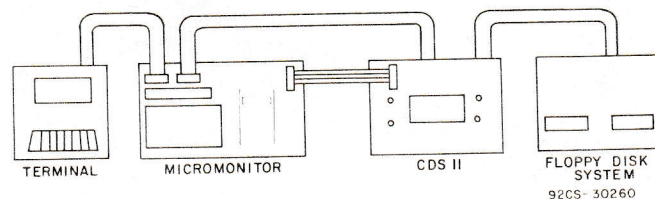


Fig. J9 - Typical developmental system setup.

12. Follow the procedure given in the RCA COSMAC Floppy Disk System II CDP18S805 Instruction Manual, MPM-217, to run the editor to create a source file and the assembler to create a listing file. A printout of the listing file will probably be needed during the debugging session; it should be obtained before proceeding further.

13. Load the listing file or the hexadecimal file of the application program into the CDS RAM using the \$L command.

14. When loading is completed, press the BREAK button on the Micromonitor to regain Micromonitor control.

15. Press SH then 10, 30 or 120, as appropriate, to enable communication between the terminal and the Micromonitor. A decimal point prompt should be printed. This step disables the built-in keyboard and automatically severs the path between the CDS and the terminal.

After the foregoing steps, the debugging operation can begin using the Micromonitor commands to set breakpoints, set and examine registers, control external signals, and others. Start the program using the \$Paaaa:h h h h command. Specify an address (normally \$P0) in order to set $X=P=0$ when starting the program. When debugging only a small section of a larger program that assumes certain values already exist in various registers, these registers may be initialized with the !R, !D, !P, and similar commands. The program may then be started by a \$P command without an address specified.

Program patches can be made by use of the !M command to modify memory. The listing printout can be used for notes of the changes that need to be made. When the program is ready for editing and re-assembly, control must be returned to UT20. Follow the procedure outline above, starting at step 9. Control can always be returned to the Micromonitor by pressing the BREAK button on the keyboard of the Micromonitor.

